DOCUMENT RESUME

ED 325 678 CE 056 259

AUTHOR Slack, Don
TITLE Microprocessor and Computer Systems. Electronics Module 8. Instructor's Guide.
INSTITUTION Missouri Univ., Columbia. Instructional Materials Lab.
SPONS AGENCY Missouri State Dept. of Elementary and Secondary Education, Jefferson City. Div. of Vocational and Adult Education.
PUB DATE 90
NOTE 302p.; For related documents, see ED 315 659-662 and CE 056 256-261.
AVAILABLE FROM Instructional Materials Laboratory, University of Missouri, 2316 Industrial Drive, Columbia, MO 65202 (Order No. 70-5008-I: $24.70, including disks).
PUB TYPE Guides - Classroom Use - Guides (For Teachers) (052)

EDRS PRICE MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS Behavioral Objectives; Competency Based Education; *Computers; *Computer System Design; *Debugging (Computers); *Electronics; Equipment Maintenance; Evaluation; Industrial Arts; Learning Activities; Learning Modules; Microcomputers; Postsecondary Education; Problem Solving; *Programing; Repair; Secondary Education; State Curriculum Guides; *Trade and Industrial Education; Troubleshooting
IDENTIFIERS Missouri

ABSTRACT

This module is the eighth of 10 modules in the competency-based electronics series. Introductory materials include a listing of competencies addressed in the module and a cross-reference table of instructional materials. Seven instructional units cover: computer architecture; operating systems; computer programming; microprocessor instructions; computer systems and address decoders; interface and data transfer techniques; and computer troubleshooting. Each unit includes some or all of the following components: performance objectives; information sheets; transparency masters; handouts; assignment sheets; job sheets; and unit test with answer key. Special instructions are included in the notes to the instructor section near the beginning of each unit. (YLB)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

70-5008-I
Instructor's Guide

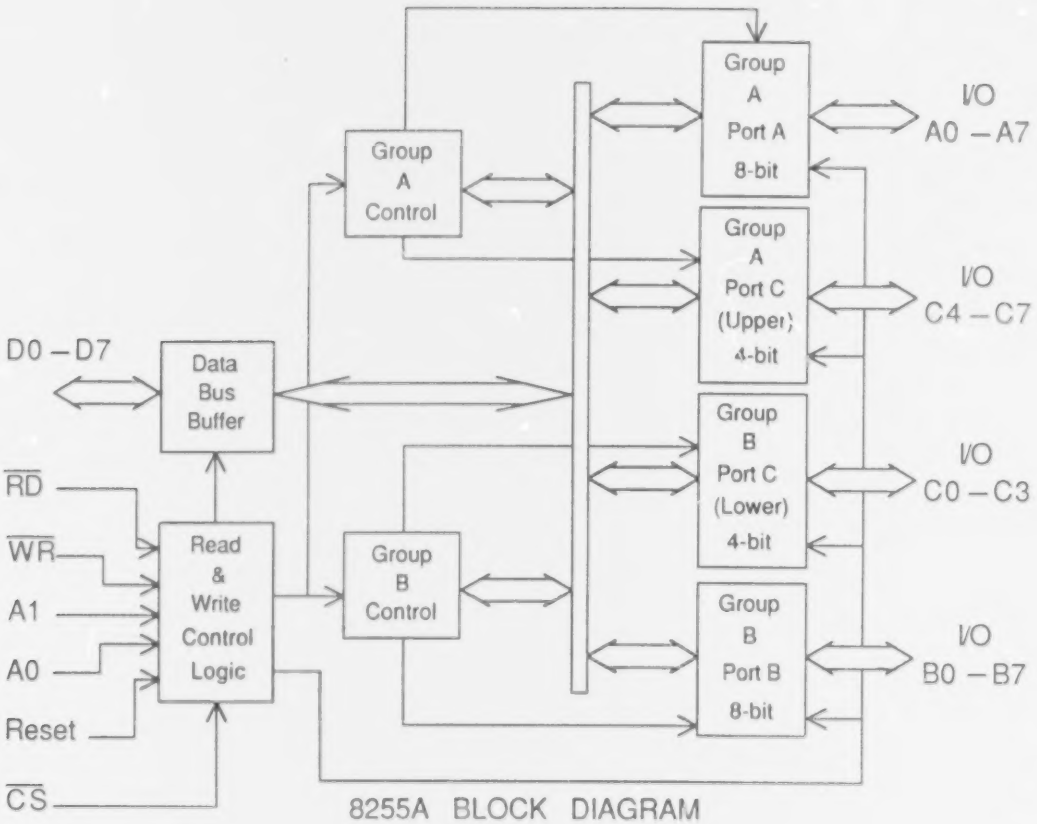
ELECTRONICS

Module

8

MICROPROCESSOR AND COMPUTER SYSTEMS

ED325678



8255A BLOCK DIAGRAM

CE 054 259

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

Minor changes have been made to improve reproduction quality.

Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Modules in this IML Electronics series

1: *Electronic Safety* by Bill Tharp
2: *Fundamentals of Electronics* by Jim Everett
3: *Digital Logic Application* by Ed Carter and Mark Murphy
4: *Circuit Construction Techniques* by Don Slack
5: *Semiconductor Devices and Applications* by John Chappell, Bill Franken and Mark Murphy
6: *Receivers and Transmitters* by Jim Everett
7: *Testing and Troubleshooting* by Don Slack
8: *Microprocessor and Computer Systems* by Don Slack
9: *Optical Electronics* by Bill Franken
10: *Electromechanical Devices and Controllers* by Ed Carter

The activity which is the subject of this report was supported in whole or in part by funds from the Department of Elementary and Secondary Education, Division of Vocational and Adult Education. However, the opinions expressed herein do not necessarily reflect the position or policies of the Missouri Department of Elementary and Secondary Education or the Division of Vocational and Adult Education, and no official endorsement should be inferred.

ELECTRONICS

Module 8: Microprocessor and Computer Systems

by

Don Slack

Diane M. Davis, editor/project coordinator
Instructional Materials Laboratory
8 London Hall
University of Missouri-Columbia
Columbia, MO 65211

1990

Instructor's Guide
Catalog No. 70-5008-I

FOREWORD

Once again, we are indebted to the teachers and administrators who provide their time, efforts and professionalism to develop curriculum for trade, technical and industrial education programs. This electronics guide is an outstanding example of what can be accomplished when the right people with the right attitude get together. Committee members worked hard over several years' time in advisory, writing, technical consultation and field-test capacities.

This guide is designed to be totally integrated with the Vocational Instructional Management System. It is hoped that all guides produced under our supervision will help classroom teachers manage instruction. A curriculum guide is more than a resource--it is an integral part of the management system. Rather than impeding instruction, it should guide and facilitate instruction.

This guide's format has been set up for competency-based teaching and testing. Unit tests evaluate specific competencies and can be used with the Vocational Administrative Management System (VAMS). Information sheets, transparency masters, job sheets, assignment sheets and test items are keyed to specific competencies. A cross-reference table has been included to quickly identify parts of the curriculum that relate to specific competencies. The material has also been packaged for students.

Although this guide may be a new format for some teachers, we are confident that, when used as designed, time spent by teachers and students will become more productive. Any future electronics curriculum modifications will begin with this guide. The electronics field is undergoing constant and considerable change. We will annually evaluate the need to update this guide on a module-by-module basis. Your suggestions regarding areas for improvement are both encouraged and appreciated.

Judith Moore, supervisor
Industrial Education
Department of Elementary and Secondary Education

Charles "Chuck" Waibel, director
Industrial Education
Department of Elementary and Secondary Education

ACKNOWLEDGMENTS

This electronics guide represents a continuing commitment to Missouri's Vocational Instructional Management System (VIMS). The guide has been keyed to the electronics competency profile developed by industry and education professionals in the state.

When searching for current electronics materials to adopt or utilize in the guide's development, comparable materials were found from the Mid-America Vocational Curriculum Consortium (MAVCC), of which Missouri is a member. MAVCC materials, therefore, have been relied upon as a resource for this guide. A debt of thanks is owed to the entire MAVCC organization for the research and development of these electronics materials.

Special appreciation is given to the writers (Ed Carter, John Chappell, Jim Everett, Bill Franken, Mark Murphy, Don Slack and Bill Tharp) for their self-motivation, reliability and technical expertise. IML employees Crystal Dietiker and Donna Curtis contributed greatly to the guide's production.

These industry representatives and educators contributed to the guide as members of the advisory committee.

Ed Carter, Columbia Area Career Center, Columbia
John Chappell, State Fair Community College, Sedalia
Jim Everett, Platte Co. Area Vocational-Technical School (AVTS), Platte City
Bill Franken, Maple Woods Community College, Kansas City
Linda Miller, St. Louis Community College at Florissant Valley
Mark Murphy, Davis H. Hart Mexico AVTS, Mexico
Don Slack, Graff Area Vocational Center, Springfield
Bill Tharp, Chillicothe AVTS
Charles Waibel and Judith Moore, Division of Vocational & Adult Education, Department of Elementary and Secondary Education, Jefferson City

Educators who field-tested the guide in the classroom include the following.

Don Campbell, Boonslick AVTS, Boonville
John Chappell, State Fair Community College, Sedalia
Bob Chenoweth, Chillicothe AVTS
Jim Everett, Platte Co. AVTS, Platte City
Don Kennison, Nichols Career Center, Jefferson City
Don Mallory, Cass County AVTS, Harrisonville
Linda Miller, St. Louis Community College at Florissant Valley
Don Slack, Graff Area Vocational Center, Springfield
Lawrence Vance, Rolla AVTS
Mark Ward, Lewis & Clark Technical School, St. Charles

Support and contributions of IML staff members were instrumental to this module's development.

Harley Schlichting, director
Amon Herd, associate director
Phyllis Miller and Dan Stapleton, assistant directors
Sharon Dyer, graphic designer
Hsiu-Nien (Shelley) Chen, student assistant - technical
Donna Curtis and Lori Holliday, word processors
Mary Beth Ramsey, student assistant - clerical

TABLE OF CONTENTS

Unit I: Computer architecture 1
Unit II: Operating systems 57
Unit III: Computer programming 93
Unit IV: Microprocessor instructions 139
Unit V: Computer systems and address decoders 229
Unit VI: Interface and data transfer techniques 277
Unit VII: Computer troubleshooting 327

COMPETENCIES ADDRESSED IN THIS MODULE

I. **Microprocessor and Computer Systems**

1. Identify and describe computer architecture
2. Identify the levels of computer languages
3. Describe the purpose of operating systems
4. Analyze simple programs using computer instruction sets
5. Write simple computer programs
6. Debug software
7. Describe interfacing techniques
8. Explain control and data transfer techniques to and from peripheral devices
9. Troubleshoot hardware problems

H. **Test Equipment**

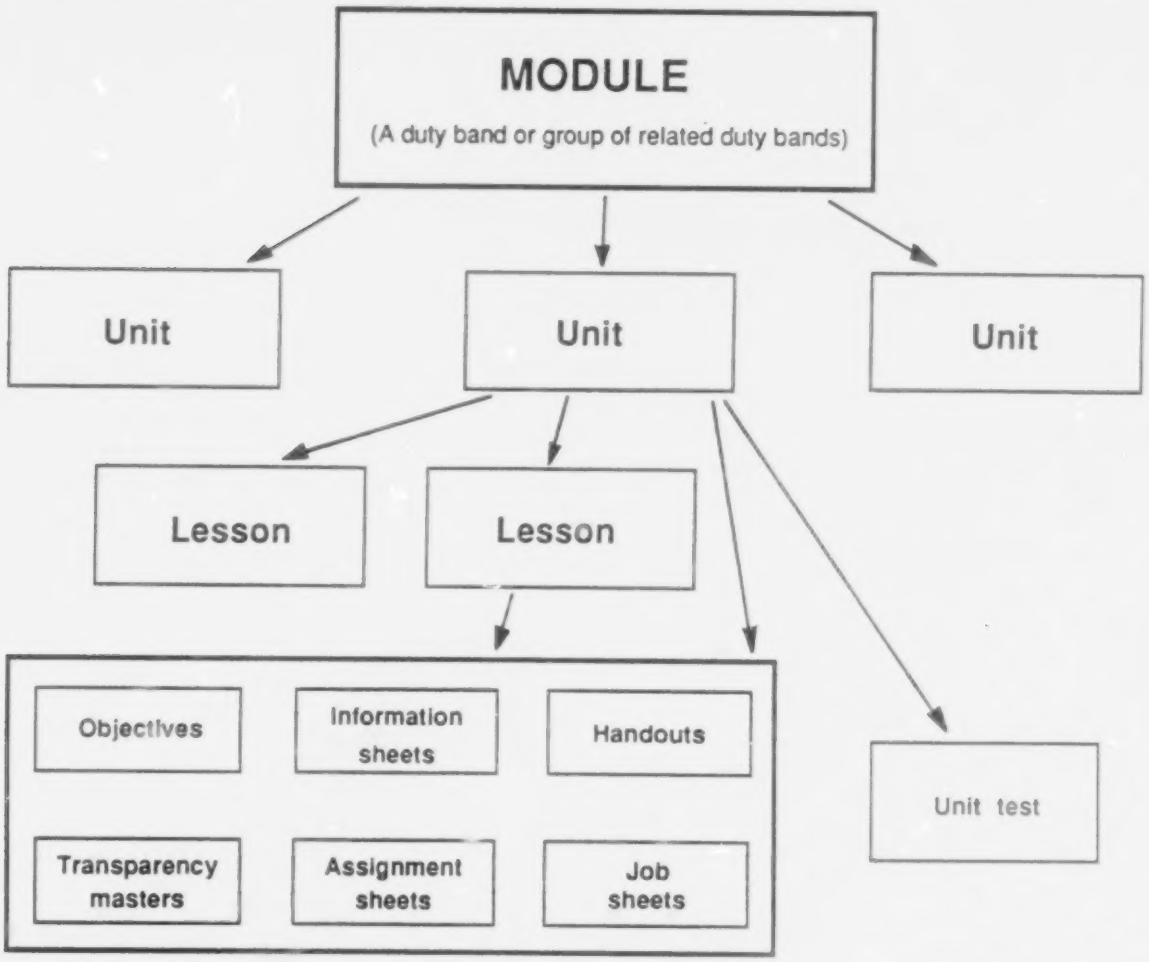
7. Describe the operation and demonstrate the use of digital breakout boxes
8. Describe the operation and demonstrate the use of analog breakout boxes

HOW TO USE THIS PUBLICATION

GUIDE COMPONENTS

Units of instruction

This Electronics guide contains ten modules of instruction. Within each module are instructional units including some or all of the following components: performance objectives, information sheets, transparency masters, handouts, assignment sheets, job sheets and a unit test. Special instructions on audiovisuals, supplies, etc. are included in the Notes to the Instructor section near the beginning of each unit.



Competency profile

On the Missouri competency profile, tasks are divided into general subject areas, or duty bands, which are identified by a capital letter. Under each duty band are related tasks (competencies), which are identified by an Arabic numeral. To evaluate the student on performance of the competencies, a 3-2-1-N grading system is used, where:

- 3 = **Mastery** - can work independently with no supervision
- 2 = **Requires supervision** - can perform job completely with limited supervision
- 1 = **Not mastered** - requires instruction and close supervision
- N = **No exposure** - no experience or knowledge in this area

Cross-reference table (color-coded white)

Once the numbering system is understood, this table quickly reveals how the instructional components of the module relate to Missouri competencies. On the competency profile, tasks are divided into general subject areas, or duty bands, which are identified by a capital letter. Under each duty band are related tasks (competencies), which are identified by an Arabic numeral.

On the cross-reference table, unit numbers are represented by Roman numerals (I, II, III, etc.). Lessons within a unit are shown with Arabic numerals. Therefore, the first objective in Lesson 1 is numbered 1.1, the second is 1.2, and so on. Other components of the module (transparencies, handouts, assignment sheets and job sheets) also use this numbering system. By looking at the cross-reference table, the instructor can quickly determine what test questions or job sheets a student needs to receive an evaluation on a particular competency.

Objectives (color-coded white)

Each unit is based on performance objectives which state the measurable unit and specific behavioral or performance objectives that students are expected to achieve. Since the objectives of the unit provide direction for the teaching-learning process, it is important for the teacher and students to have a common understanding of the objectives' intent. Teachers should feel free to add/adapt objectives which will fit the material to student and community needs. Supporting material will, of course, need to be added for any new objectives.

Table of contents, references, notes to the instructor (color-coded pink)

These pages have been coded pink for instructor use only. The reference section lists materials used as sources in developing the material. Special instructions on audiovisuals, supplies, etc. are included in the Notes to the Instructor section.

Information sheets (green)

Presented in outline form for clarity, these pages provide content essential for meeting the cognitive (knowledge) objectives in the unit. Students should study the information sheets before class discussion or completion of assignment sheets. **CAUTION** statements indicate possible equipment damage, while **WARNING** statements recommend procedures to avoid bodily injury. Handouts and transparency masters have been keyed in the information sheets for instructional use.

Transparency masters (white)

Transparencies provide information by visually reinforcing or clarifying ideas presented in the information sheets. They are effective when identification is necessary and direct the class's attention to the topic of discussion. Transparencies should be left on the screen only when topics shown are under discussion, however.

Handouts (white)

Whenever more detailed information than can be presented in a transparency is needed, a handout is used. Handouts give the students additional information that can be kept with their class notes or student references.

Assignment sheets (tan)

Assignment sheets furnish paper-and-pencil practice to develop the knowledge necessary as a prerequisite to skill development. They can be given to students for class completion or used as homework assignments. Answer sheets (color-coded pink) can be used by the student or instructor to check progress.

Job sheets (blue)

These evaluation instruments are provided for two purposes: to outline steps of procedure and to check student performance mastery. Job sheets begin with a list of tools and materials needed for the activity. Job sheets furnish potential employers with a picture of skills being taught and performance which might

be reasonably expected from a vocational student. The spaces to the right of the page on a job sheet assist the instructor in evaluating a student on a particular activity. This 3-2-1-N grading system is the same as the one used on the competency profile.

Unit test (yellow)

Competency-based tests are designed to help eliminate subjective testing and to improve student evaluation and accountability. Specific competencies and test questions are linked on the answer key. These unit tests have been written to be used with computerized VAMS (Vocational Administrative Management System), utilizing Scantron sheets for computer-recording of test answers.

The unit test may be eliminated if the competency was thoroughly evaluated by job sheets. However, an emphasis should be placed on cognitive as well as performance skills. If the objective or competency will be evaluated by a job sheet, it is often necessary to determine by a written test whether the student understands enough of the material to perform the job sheet.

SUGGESTED ACTIVITIES

The following list of suggested activities may be helpful, especially to the beginning instructor.

1. Provide students with objective sheet and information sheets.
2. Discuss unit and specific objectives.
3. Discuss information sheets, assignment sheets and/or job sheets, lesson-by-lesson.
4. Have students do assignment sheet(s) to satisfactory levels.
5. Give students unit test.
6. After successful completion of the test, have students do job sheet(s).

Students should perform the following activities.

1. Read objective sheet.
2. Study information sheets.
3. Do assignment sheet(s).
4. Take unit test.
5. Do job sheets.

PACKAGING

This Electronics guide has been packaged with the entire class in mind. For the teacher, a complete instructor's guide contains all ten modules. In the student reference, selected portions have been reproduced for studen: use (objectives page, information sheets, handouts). Student work sheets (assignment sheets, job sheets, unit tests) have been packaged together for student distribution as the instructor deems necessary.

In the student reference, page numbers do not run consecutively because some pages have been removed that appear only in the instructor's guide.

PARTS/EQUIPMENT LIST

BASIC language
BASIC program written during JS 2.1-2.4 (Unit III)
Diskettes
DOS master diskette with DEBUG.COM
GWBASIC diskette (or BASICA)
IBM-PC or compatible computer
MS-DOS or PC-DOS

MODULE 8: MICROPROCESSOR AND COMPUTER SYSTEMS

CROSS-REFERENCE TABLE OF INSTRUCTIONAL MATERIALS

Compe-tency	Instructional Objective	Transparency Master	Handout	Assignment Sheet	Job Sheet	Test Item	Notes
I1	I 1.1-3.4	I 2.1-2.3	I 3.1-3.2	I 1.1, 2.1	I 3.1	I 1-39	
I2	III 1.1-1.3			III 1.1		III 3-4, 7-9	
I3	II 1.1-2.4			II 1.1	II 2.1	II 1-44	
I4-6	IV 3.1-5.4			IV 3.1	IV 3.1-5.2		
I5	III 2.1-2.4, IV 1.1-2.2, VI 1.4-1.6		IV 1.1, 1.2, 2.1		III 2.1-2.4, IV 1.1, 2.1, VI 1.2-1.4	III 1-2, 5-6, 10-11, 13-19	
I6	III 2.5				III 2.5		
I7	V 2.1-2.4, VI 1.1-1.6, 2.8-2.12	V 2.1-2.3, VI 1.1-1.2	V 2.1, VI 1.1, 1.2		V 2.1, VI 1.1-1.4, 2.1	V 11-17, 19-20, VI 7, 27	
I8	VI 1.3, 2.1-2.7	VI 2.1	VI 2.1, 2.2		VI 1.1	VI 1-5, 8-26	
I9	V 1.2-1.6, VII 1.1-1.8		V 1.1, VII 1.1-1.3	V 1.1	V 1.1-1.2, VII 1.1-1.3	V 1-10	
H7-8	V 1.1-1.6, VI 2.11				V 1.1-1.2, VI 2.1	V 18, VI 8	

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT I: COMPUTER ARCHITECTURE

UNIT OBJECTIVE

Upon completion of this unit, the student should be able to describe computer architecture and define computer and microprocessor technical terms. The student should also be able to list the milestones in computer evolution and list common microprocessor families. This knowledge will be evidenced by achieving the specified score on the job [illegible]nd unit test.

SPE[illegible]IC OBJECTIVES

After completion of the lessons in this unit the student should be able to:

1.1 Explain the evolution of microcomputers.

1.2 Explain the differences between the 8086 and 8088.

1.3 Explain which microprocessor is used in popular microcomputers.

2.1 Explain the internal architecture of a generic microprocessor and typical computer.

2.2 Explain the operation of the fetch/execute cycle.

2.3 Define RAM, ROM, peripheral, I/O, ALU, accumulator, register.

2.4 Explain the functions of the three major sections of a microprocessor.

2.5 Explain the functions of the three major sections of a microcomputer.

2.6 Define MPU, CPU, program counter, bus and interface.

2.7 Define op code, operand, address, data and mnemonic.

3.1 Explain the function of the internal architecture of an 8086/88 microprocessor. (competency I1)

3.2 Define stack, IP and segment.

3.3 Explain where data can be relocated using the 8088 MOV instruction.

3.4 Display and change contents of registers with Debug.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT I: COMPUTER ARCHITECTURE

CONTENTS OF THIS UNIT

A. Objective sheet

B. References

C. Notes to the instructor

D. Lesson plans

1. Lesson 1: HISTORY OF THE MICROPROCESSOR

   a. Information outline

   b. Assignment sheet

      AS 1.1: Evolution of the microprocessor

   c. Answers to assignment sheet

2. Lesson 2: TYPICAL COMPUTERS

   a. Information outline

   b. Transparency masters

      TM 2.1: Simplified microcomputer block diagram
      TM 2.2: System architecture
      TM 2.3: Generic microprocessor

   c. Assignment sheet

      AS 2.1: Microprocessor fundamentals

   d. Answers to assignment sheet

3. Lesson 3: 8088 ARCHITECTURE

   a. Information outline

   b. Handouts

      HO 3.1: The 8086/8088 CPU
      HO 3.2: Debug commands

   c. Job sheet

      JS 3.1: Practice with Debug and the 8088

E. Unit test

F. Answers to unit test

REFERENCES

A. Books

1. Brey, Barry B. Microprocessors and Peripherals: Hardware, Software, Interfacing and Applications. Columbus, OH: Merrill, 1988.

2. Brey, Barry B. The Z80 Microprocessor: Hardware, Software Programming and Interfacing. Englewood Cliffs, NJ: Prentice-Hall, 1988.

3. Gillmore, Charles M. Microprocessors: Principles and Applications. New York: McGraw-Hill, 1989.

4. Hall, Douglas V. Microprocessors and Interfacing: Programming and Hardware. New York: McGraw-Hill, 1986.

5. Harman, Thomas L. The Motorola MC68020 and 68030 Microprocessors: Assembly Language, Interfacing, and Design. Englewood Cliffs, NJ: Prentice-Hall, 1989.

6. Hughes, Fredrick W. Microprocessor Technology: Theory and Experimentation. Englewood Cliffs, NJ: Prentice-Hall, 1989.

7. Miller, Michael A. The 68000 Microprocessor: Architecture, Programming, and Applications. Columbus, OH: Merrill, 1988.

8. Norton, Peter. Inside the IBM PC. New York: Brady, 1986.

9. O'Connor, Patrick Joseph. Digital and Microprocessor Technology. 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1989.

10. Pasahow, Edward J. Microprocessor Technology and Microcomputers. New York: McGraw-Hill, 1988.

11. Singh, Avtar, and Walter A. Triebel. The 8088 Microprocessor: Programming, Interfacing, Software, Hardware, and Applications. Englewood Cliffs, NJ: Prentice-Hall, 1989.

12. Tocci, Ronald J., and Lester P. Laskowski. Microprocessors and Microcomputers: Hardware and Software. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1987.

B. Curriculum guides

1. Brown, A. O. Advanced Microcomputer Service Technician. Stillwater, OK: Mid-America Vocational Curriculum Consortium, 1985.

2. Vorderstrasse, Ron, and Leo Siebert. Communications Electronics. Stillwater, OK: Mid-America Vocational Curriculum Consortium, 1988.

NOTES TO THE INSTRUCTOR

A. Suggested approaches to teaching this unit

1. Discuss history of microprocessor development.

2. Discuss a generic microprocessor's architecture.

3. Discuss registers, busses and instruction pointers.

4. Discuss the fetch/execute cycle.

5. Explain step-by-step execution of simple program.

6. If an IBM-compatible computer is available, make back-up copies of the program disk and print out READ.ME file by using the DOS command COPY READ.ME PRN.

7. Answers to JS 3.1

   a. Procedure 1

      i. (Step 3) 0003 0001 0119 FF1B 000A 0006 0016 0006 0017 0003

      ii. (Step 4) 0003 -1 Instead of -1, Debug shows 1-2=FFFF. FFFF is twos complement of -1.

      iii. (Step 5) -1 in binary would be 0000 0000 0000 0001

Complement each digit	1111 1111 1111 1110
Then add 1	1
The sum is	1111 1111 1111 1111
Converted to hex	F F F F

      Yes, Debug gave FFFF as the answer to 1-2.

   b. Procedure 3

      I. (Step 5b) AX=0000 BX=0000 CX=0000 DX=0000 IP=0100

   c. Procedure 4

      i. (Step 5b) The segment address will be the same as before.

      ii. (Step 5c) No

      iii. (Step 5d) 0200

      iv. (Step 5e) Yes

      v. (Step 5f) Yes

   d. Procedure 5

      i. (Step 10c) XXXX : 0100 (Segment address XXXX depends upon how much memory is in the PC in use)

B. Suggested student activities

1. After reading the material, do the appropriate assignment sheets.

2. Work the CPU tutorial program on disk.

3. If an IBM-compatible computer is available, do the job sheet.

4. Take the unit test.

C. Suggested supplemental material

1. Current microprocessor data manuals

2. Microprocessor Fundamentals (video, E-87-340). Sedalia, MO: The Media Center, State Fair Community College, 1987. (Available for purchase from SFCC; available for loan from Missouri Vocational Resource Center, phone 800/392-7217 or 314/882-9610)

3. Demonstration disk

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT I: COMPUTER ARCHITECTURE

LESSON 1: HISTORY OF THE MICROPROCESSOR

I. Terms and definitions

A. Byte--a single group of usually eight bits of information

B. CPU (central processing unit)--the "brain" of a computer consisting of an arithmetic logic unit, registers, timing and decoding circuitry

C. Mainframe computer--large business or scientific computer, usually a 64-bit CPU, capable of fast data processing with many multiple user terminals; contains massive memory and input/output facilities

D. Microcomputer--desk-top sized computer; microprocessor CPU, usually an 8-bit or 16-bit CPU, with memory and input/output facilities

E. Microprocessor--The CPU consists of an arithmetic logic unit, registers, timing and decoding circuitry, usually contained in a single integrated circuit that controls computer activity.

F. Minicomputer--medium-sized business or scientific computer, usually a 32-bit CPU, capable of fast data processing with several user terminals; contains memory and input/output facilities

G. Supercomputer--large, extremely fast computer

II. Evolution of the computer

A. World's first digital computer constructed during World War II

1. Thousands of electromechanical relays

2. Comparable now to a cheap calculator

3. Slow and unreliable

B. The first vacuum tube digital computer constructed during World War II in 1943

1. University of Pennsylvania

2. Thousands of vacuum tubes

3. Required tons of cooling air

4. Slow and unreliable

C. Second generation used solid-state transistors (1947)

1. More than 30,000 transistors in the CPU

2. Less heat generated

a. Still required massive amounts of cooling air

b. More reliable due to transistors

D. Third generation used the integrated circuit (1958)

1. Made the CPU smaller, generated less heat

2. CPU could require more than a thousand I.C.s

III. The microprocessor

A. In 1969, Intel announced the world's first microprocessor, the 4004, a 4-bit CPU.

1. Designed to replace logic circuits of a calculator

2. An integrated circuit

3. All the logic circuitry for a CPU

4. Less than about 1/4" square of silicon

5. MPU = microprocessor CPU

B. In 1970 Intel released an 8-bit microprocessor (8008 model).

1. Designed to be the heart of a CRT terminal

2. Had a small, simple instruction set

3. Could address only 16K of memory.

C. In 1973 Intel released the first microprocessor powerful enough for a general-purpose computer.

1. Though an 8-bit MPU, the 8080 had an extensive instructions set.

2. The 8080 could address 64K of memory.

IV. The first microcomputers

A. The first personal computers were built with the 8080.

1. Altair offered a personal computer as a kit in 1975.

a. First appeared as a magazine construction article

b. Programmed with front panel toggle switches; binary codes grouped in octal form

c. Data read out on front panel; binary LEDs grouped in octal form

d. 4K or 8K memory boards

2. Imsai offered a personal computer as a kit or factory wired.

a. Marketed a few months after the Altair

b. The Imsai featured in the movie "War Games"

3. Within a few months, many home and small business computers were for sale.

   a. Many boards and computers manufactured by individuals in home basements or garages and new, small companies and corporations

   b. Sold through stores and by mail order

   c. Sold wired or as kits

B. Other companies introduced new microprocessors.

1. Motorola introduced the MC6800 microprocessor, which is used in automobiles, appliances and industrial controls.

2. Commodore Business Machines introduced the 6502.

   a. Bought a small company that designed the 6502

   b. 6502 family still used in their 8-bit microcomputers

3. Zylog introduced the Z80 microprocessor.

   a. Compatible with the 8080

   b. Faster than 8080

   c. Has an extended instruction set

   d. Used by Radio Shack in their TRS-80 Model 1; was the first low-cost home computer

4. Intel followed with the 8085, an improved, faster version of the 8080.

C. Fortunes were made and corporations were born as young programmers wrote programs for their own use, then found a market for their product.

1. BASIC

   a. Sold it to Altair and Imsai owners on cassette tape

   b. Grew into the giant Microsoft Corporation

   c. Microsoft BASIC used on most microcomputers

2. 8" floppy disk

   a. No disk operating system was available to do so.

   b. A young man wrote his own operating system called CP/M.

   c. His company grew into Digital Research.

D. More than a hundred companies built small business computers.

1. Most small business computers used the 8085 or Z80.

2. Most 8085/Z80s used the CP/M operating system.

V. The new generation of microcomputers

A. In 1978 Intel surprised the microprocessor world with a 16-bit MPU, the 8086.

1. The 8086 could handle 16-bits of data in internal registers and on external bus, at one time.

2. Most support chips and devices available at that time were 8-bit.

3. Intel also released the 8088 to use existing 8-bit hardware.

B. In the fall of 1981, IBM entered the microcomputer market with a machine based on the 8088.

1. Probably to keep the basic system cost down

2. Used readily available 8-bit support chips

3. Small business computer market soon dominated by IBM

C. Competitors entered the market with IBM-compatible computers.

1. Most were built around the 8086 microprocessor.

2. 8086 has a slight processing speed advantage over 8088.

3. Other companies include Sun, Hewlett-Packard, Digital Electronics Corporation, Wang, Prime and Tandon.

D. IBM PS/2 line replaced the original PC line.

1. PS/2 uses 8086 in low-end models.

2. High-end models use newer generation microprocessors.

E. The 8086 family of microprocessors is versatile.

1. 8088--8-bit external bus and 16-bit internal register

2. 8086--16-bit external bus and 16-bit internal register

3. 80186

a. 16-bit external bus

b. 16-bit internal register

c. Faster than 8086/8088

d. Extended instruction set

e. Upward compatible with 8086/88

4. 80286

a. 16-bit external bus

b. 16-bit internal register

c. Faster than 8086/8088 or 80186

d. Extended construction set

e. Upward compatible with 8086/88, 80186

f. Can be multi-tasking and multi-user

5. 80386

a. Can be faster than 80286

b. Extended construction set

c. 16/32-bit

d. Can access more memory

e. Can be multi-tasking and multi-user

6. 80486

a. A "supercomputer" on a chip

b. Includes math coprocessor on CPU chip

c. First available in late 1989

d. Extremely powerful and fast

INSTRUCTOR'S NOTE: Check current microprocessor data manuals for the latest developments.

F. Additional microprocessor families include the following.

1. 6800 family

a. 6809--8/16-bit used in Tandy's color computer

b. 68000--32-bit used in Amiga, Macintosh and Atari ST

2. 6502 family--8-bit used in Apple, Commodore and Atari

3. Other microprocessors used primarily for industrial controls

EVOLUTION OF THE MICROPROCESSOR

Define the following terms.

1. Microprocessor--

2. Minicomputer--

3. Mainframe computer--

4. Microcomputer--

5. CPU--

Answer the following questions with short answers.

6. Which company invented the microprocessor?
7. Which 8-bit microprocessor was most used for small business computers?
8. Which 8-bit microprocessor is most used for home/game type computers?
9. Which microprocessor is used in the IBM PC/XT?
10. Which microprocessor is used in the Apple II series?
11. Which microprocessor is used in the Commodore 64?
12. Which microprocessor is used in the Macintosh?
13. Which microprocessor is used in the Amiga?
14. What is the difference between the 8088 and the 8086 microprocessors?

ANSWERS TO AS 1.1

1. The CPU consisting of an arithmetic logic unit, registers, timing and decoding circuitry usually contained in a single integrated circuit that controls computer activity

2. Medium-sized business or scientific computer, usually a 32-bit CPU, capable of fast data processing with several user terminals, and contains memory and input/output facilities

3. Large business or scientific computer, usually a 64-bit CPU, capable of fast data processing with many multiple user terminals, and contains massive memory and input/output facilities

4. Desk-top sized computer; CPU is a microprocessor, usually an 8-bit or 16-bit CPU, and contains memory and input/output facilities

5. Central processing unit, the "brain" of a computer, consisting of an arithmetic logic unit, registers, timing and decoding circuitry

6. Intel

7. Z-80

8. 6502

9. 8088

10. 6502

11. 6502

12. 68000

13. 68000

14. The 8088 has an 8-bit external bus and a 16-bit internal register, while the 8086 has both a 16-bit external bus and a 16-bit internal register

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT I: COMPUTER ARCHITECTURE

LESSON 2: TYPICAL COMPUTERS

NOTE: This unit addresses the three basic components of a computer and the function of each.

I. Terms and definitions

A. Interface--point where two components connect within a system and/or the device used to make the connection

B. Microcomputer--This computer system's processing unit is a microprocessor and contains memory and input/output facilities.

C. Microprocessor--the arithmetic logic unit, registers, timing and decoding circuitry usually contained in a single integrated circuit that controls computer activity

D. Peripheral--any device such as a disk drive, printer, modem or video display added to a microcomputer to provide increased capacity for handling, storing or presenting data

E. Register--temporary storage location for data inside the microprocessor, usually either 8-bits or 16-bits wide

II. The simplified microcomputer **(TM 2.1, 2.2)**

A. Three sections to a typical computer: central processing unit (CPU), memory and input/output ports (I/O)

1. Computers can be tailored for a specific use, e.g., word processing; accounting; process control; environment, automotive, appliance, or robotic control; arcade video games or office machines.

2. Computers can be for general-purpose use, as well. (word processing; accounting; data base; process, environment or robotic control; video games)

B. Central processing unit

1. The brain of the computer that controls everything happening in the computer and makes all decisions

2. CPU performs three main functions: control, calculation and decision

C. Memory

1. Two general types: RAM and ROM

2. RAM = data memory

a. Random-access memory: any location used in any order

b. Volatile memory: all information lost when power is turned off

c. Read, get information from RAM

d. Write, store information in RAM

3. ROM = program memory

a. In read-only memory, any location can be used in any order and accessed randomly.

b. This non-volatile memory holds all information when power is turned off, permanently storing information.

c. While ROM memory can be read from, it cannot be written on by users.

4. Stores computer instructions and the data to be operated upon

5. Address

a. Location of the memory cell

b. Similar to a row of letter boxes

c. Location or box number is the address

6. Data

a. Contents of the address

b. Similar to a row of letter boxes

c. Contents of the box is data or information

d. Data can represent instructions or information

D. I/O (input/output)

1. I/O connects the CPU to the outside world.

a. The human operator

b. Peripheral devices (disk drives, joy sticks, printers, modems and others)

c. The real world (temperature sensors, pressure sensors, motors and valves)

2. I/O section interfaces the digital CPU to peripheral analog and digital devices, as well as real-world analog and digital devices.

E. Busses **(TM 2.3)**

1. Carry information to and from sections of the computer

a. Wires

b. Foil runs on printed circuit boards

2. Three types of busses

a. Data bus carries information.

b. Address bus carries memory location or address.

c. Control bus carries control signals to and from the CPU.

F. Additional sections

1. System clock times or synchronizes the computer

2. Read/write control lines

3. CPU = the point of reference in the computer

a. Read to the CPU from memory; data copied into the CPU

b. Write from the CPU to memory; data copied into a memory location

III. Sections of the generic microprocessor CPU **(TM 2.3)**

A. CPU's three major components

1. Registers--temporary storage of binary numbers within the CPU

2. Control unit--keeps everything working together in the proper sequence

3. Arithmetic logic unit or ALU--performs the calculations

B. Registers

1. Most CPUs have several registers. Some are general-purposed, while some are for specialized purposes.

2. All CPUs have at least one main register, the accumulator. In general, all numbers processed by the CPU pass through the accumulator.

C. Control unit

1. A generic microprocessor's control unit is made of the bus control logic, instruction decoder and instruction pointer (also called program counter).

2. The bus control logic connects the CPU to data and address bus lines. It determines whether information is coming into or going out of the CPU.

3. If incoming information is an instruction or op code, the op code is passed to the instruction decoder. Then, the op code is decoded, telling the CPU what to do next.

4. If the incoming information is data (or operand), it passes data to the accumulator.

5. The instruction pointer (IP) is a counter register.

a. It holds (or points to) the address of the next location in memory or the next instruction to perform.

b. The IP always points to the next byte the microprocessor will use.

c. To change and move to a different location, a new number is loaded into the IP, which controls the direction of program flow.

D. Arithmetic logic unit (ALU)--where the actual number-crunching takes place

1. A number can be moved from accumulator to ALU or vice versa.

2. The ALU moves the second number into the accumulator.

   a. Number 4

   b. From address 0003

   c. Takes place of 3 in accumulator

   d. 4 also remains at address 0003

3. The ALU can add the two numbers together.

   a. 3 + 4 = 7

   b. 7 stored in the ALU

   c. 7 takes the place of previous number 3

4. The ALU moves numbers to the accumulator.

   a. 7 is stored in the accumulator

   b. 7 takes the place of previous number 4

   c. 7 also remains in the ALU

5. The ALU moves the number in the accumulator to address 0006.

   a. 7 is stored at address 0006

   b. 7 replaces any number that was in 0006

   c. 7 also remains in ALU and accumulator

IV. The fetch/execute cycle

A. Instructions (op codes) are stored in a computer's memory. The memory location(s) following the op code can hold the data to be operated upon (operand) and the address of where to find the operand.

B. To get these numbers from memory into the CPU, the computer follows a very precise sequence of steps called the fetch/execute cycle.

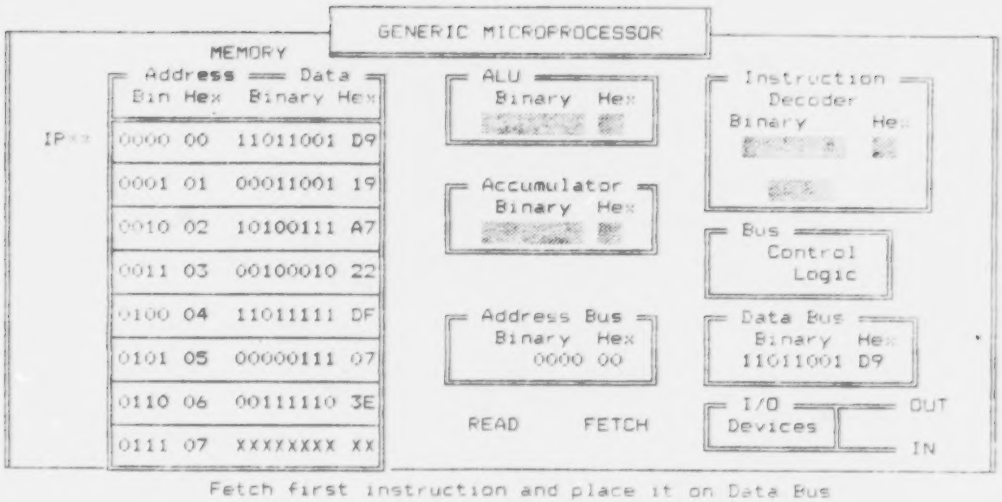
1. FETCH brings the op code number to the CPU.

2. EXECUTE performs the operation as instructed.

C. A simple program is shown on the next page.

1. Starts at address location 0000

2. Ends at address location 0006



Fetch first instruction and place it on Data Bus

3. Address and Data are shown in both binary and hexadecimal form

D. Study a generic microprocessor demonstration program.

1. The demonstration program with this unit runs on IBM-PCs or compatibles.
2. To start the program, enter either GODEMO or SDRUN MPU; press enter.
3. The previous illustration shows the first screen.
4. The screen number is at upper left corner of screen.
5. The following description of the program will refer to the screen number.

E. The fetch cycle gets the instruction information from memory and brings it into the CPU.

1. Recall that the instruction pointer points to the next location to be used.
2. Assume that the IP is pointing to memory address 0000 at the start of the program listing **(Screen 1)**.
3. Address 0000 is enabled for a read operation, and the contents of the address, $D9_H$, are placed on the data bus.
4. Since this is the first code in the program, the number represents an op code or instruction.
5. The bus control logic passes the binary number to the instruction decoder. **(Screen 2)**
6. The instruction decoder interprets the binary number to mean "take the number found in the next memory location and store it in the accumulator".
7. The IP is incremented by 1 and now points to 0001, the address of the next memory location.

F. The instruction is **executed** by enabling the memory cell now on the address bus (0001) for a read operation. **(Screen 3)**

1. Contents of address 0001, the number 19_H, is placed on the data bus.
2. The bus control logic understands that this number is an operand and moves the 19_H to the accumulator instead of the instruction decoder. **(Screen 4)**
3. The IP is incremented by one and now points to the address of the next instruction.

G. The fetch repeats.

1. The IP now points to 0002.
2. Address cell 0002 is now enabled for a read operation and the contents of the cell placed on the data bus. **(Screen 5)**
3. The bus control logic understands that it just finished handling an operand, so this number, $A7_H$, must be an op code.
4. The binary number is passed to the instruction decoder. **(Screen 6)**
5. The number is decoded as an instruction to "take the number found in the next memory location and add it to the number already in the accumulator, then leave the answer in the accumulator".
6. The instruction pointer is incremented and now points to memory location 0003.
7. Address 0003 holds the operand 22_H.

H. Execute repeats.

1. Address 0003 is enabled for a read operation and the contents, the number 22_H, is placed on the data bus. **(Screen 7)**
2. The bus control logic passes 22_H to the ALU. **(Screen 8)**
3. The IP increments to the next address.
4. The microprocessor gets ready to perform ADD instruction. **(Screen 9)**
5. The ALU adds its contents, 22_H, with the contents of the accumulator, 19_H. **(Screen 10)**
6. Addition is done in the ALU.
7. The ALU now holds the sum of $22_H + 19_H$, or $3B_H$.
8. The number $3B_H$ is copied to the accumulator, replacing the $19H_H$ that the accumulator held. **(Screen 11)**

I. Fetch repeats a third time.

1. The IP now points to address 0004.
2. Address 0004 is enabled for a read operation, and the contents placed on the data bus. **(Screen 12)**

3. Again, since the last number was an operand, the bus control logic moves the new number on the data bus to the instruction decoder. **(Screen 13)**

4. That number is decoded to be "take the number now in the accumulator and store it at memory location address found in the next operand".

5. The IP is incremented and now points to address 0005.

6. The contents of address 0005 is read and placed on the data bus. **(Screen 14)**

7. It is determined that the conter' of the accumulator will be stored at address 0007. **(Screen 15)**

J. Execute repeats.

1. Address 0007 is placed on the address bus; that location now contains random numbers. **(Screen 16)**

2. Contents of the accumulator, $3B_H$, is placed on the data bus. **(Screen 17)**

3. Memory cell 0007 is enabled for a write operation. **(Screen 18)**

4. The contents of the data bus are written to address 0007.

5. Memory location 0007 reads the number on the data bus and latches it into memory. Memory 0007 now holds the number $3B_H$.

K. Fetch repeats a fourth time.

1. The IP is pointing to address 0006, which holds a one-byte instruction.

2. Memory cell 0006 is enabled for a read operation, placing the contents on the data bus **(Screen 19)**

3. The IP is incremented by 1, pointing to address 0007. **(Screen 20)**

4. Bus control logic transfers the op code to the instruction decoder.

5. The instruction decoder determines that this op code means to "stop all execution--do not read instructions in memory locations beyond this point".

L. Execute concludes the cycle.

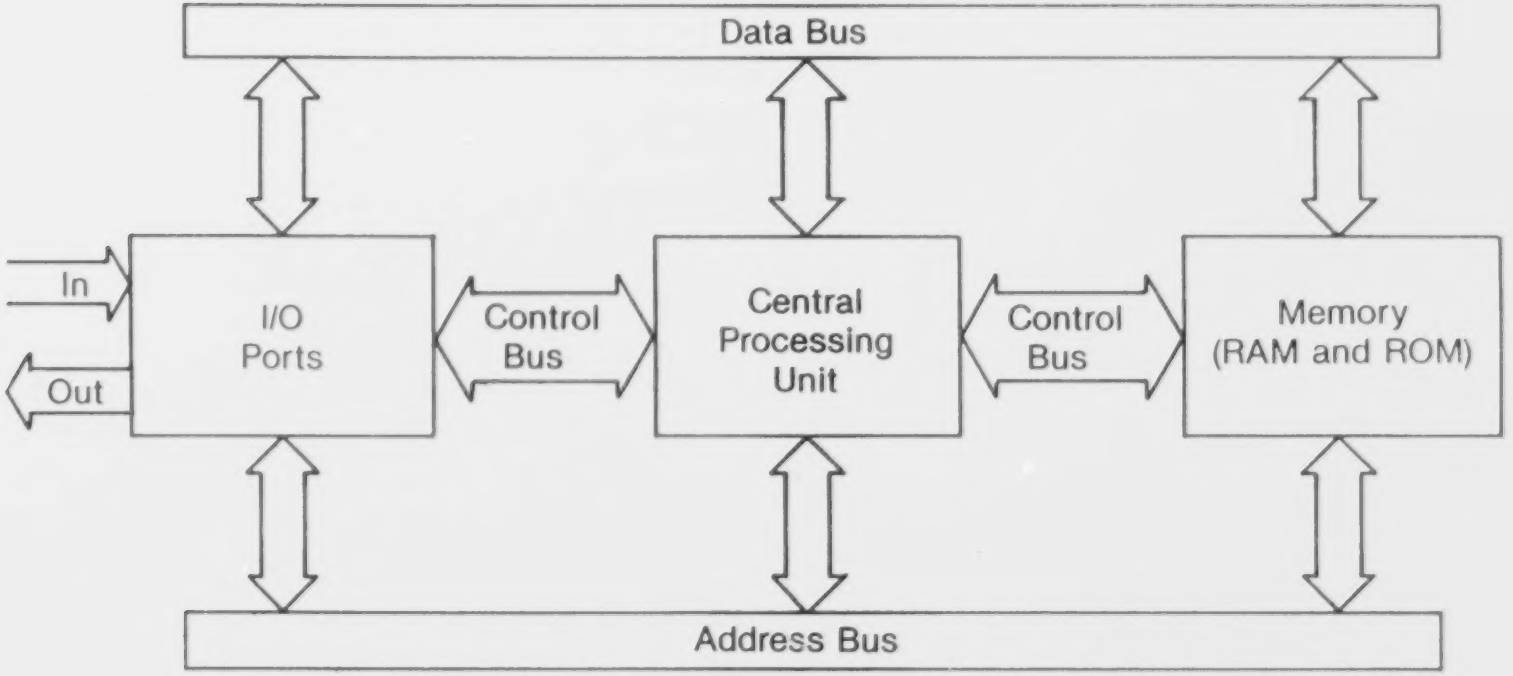
1. The microprocessor halts and the program is finished. **(Screen 21)**

2. If a halt instruction were not at the end of the program, the microprocessor would continue fetching numbers from memory.

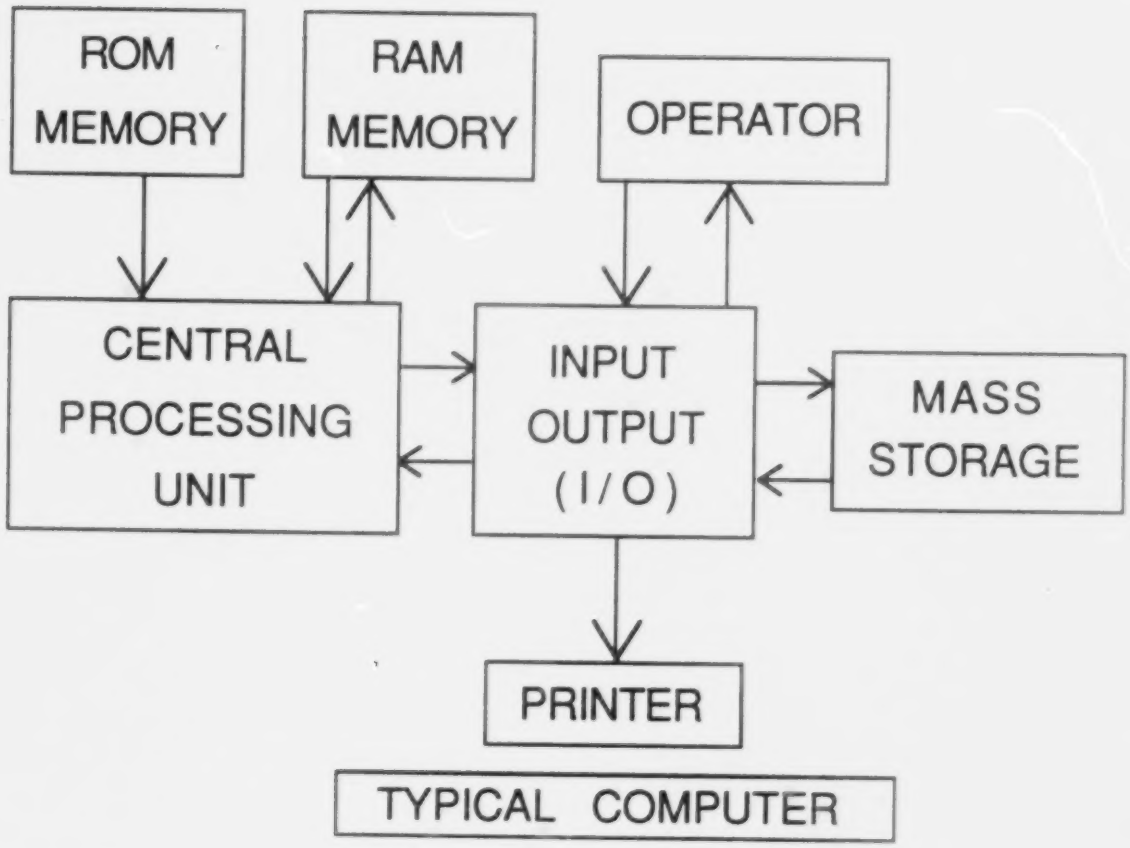
   a. The microprocessor would try to execute "garbage" numbers.

   b. The computer would lock up and lose control.

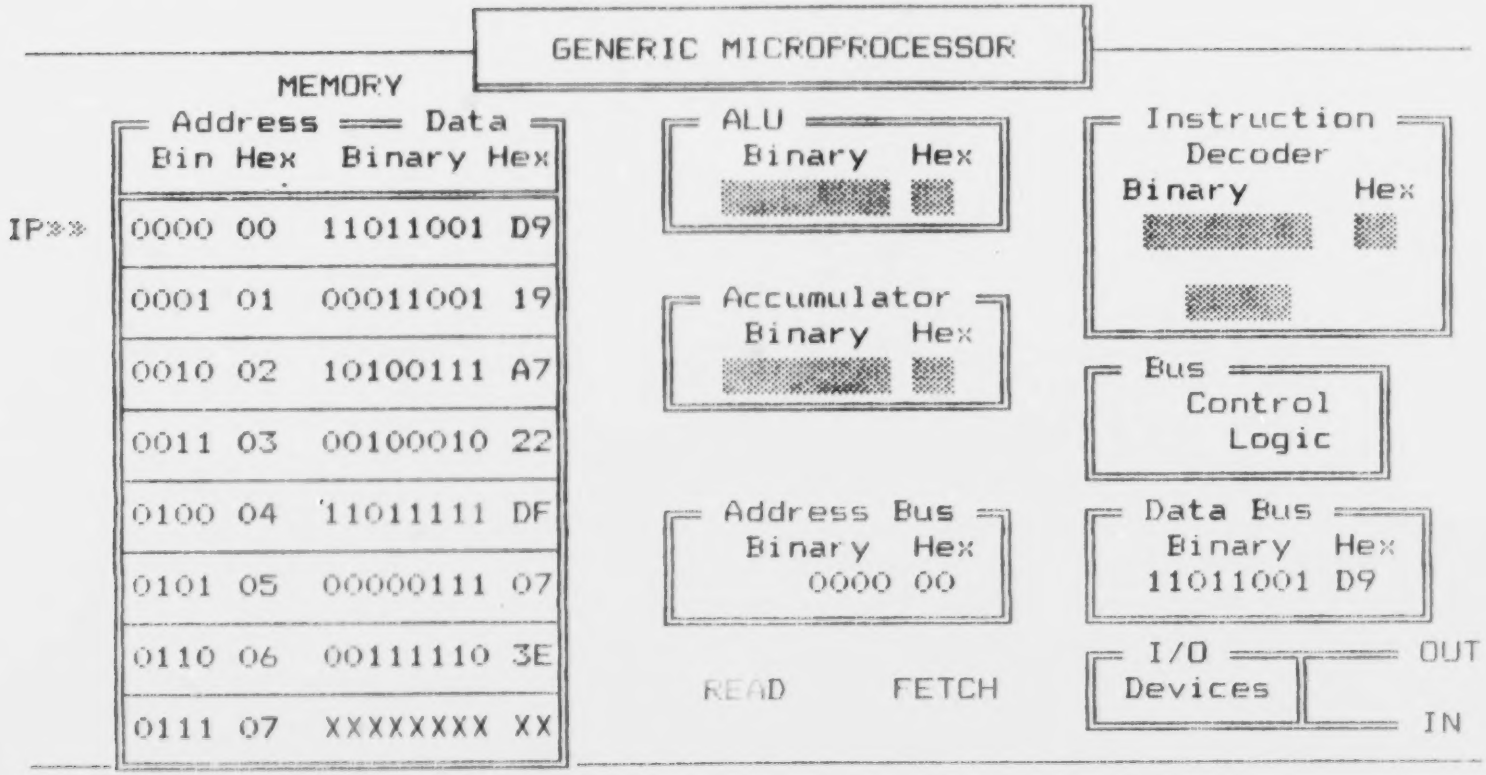
Simplified Microcomputer Block Diagram



SYSTEM ARCHITECTURE



GENERIC MICROPROCESSOR



Fetch first instruction and place it on Data Bus

MICROPROCESSOR FUNDAMENTALS

1. List the three sections of a typical computer.

   a.

   b.

   c.

2. List the two general types of memory.

   a.

   b.

3. What are the two important numbers associated with memory?

   a.

   b.

4. Which section of the computer connects the CPU to the outside world?

5. Which section is the brain of the computer?

6. What are the three main functions of the CPU?

   a.

   b.

   c.

7. What is a bus?

8. List the three types of busses and explain what they do.

   a.

   b.

   c.

9. What is a register?

10. What is an accumulator?

11. What is the function of the IP or program counter?

12. What is the function of the ALU?

13. Explain the fetch/execute cycle.

14. Explain why a halt instruction must be given to the microprocessor.

15. What is an op code?

16. What is an operand?

ANSWERS TO AS 2.1

1. a. Central processing unit (CPU)
   b. Memory
   c. Input/output ports (I/O)

2. RAM (random-access memory) and ROM (read-only memory)

3. Address and data

4. I/O (input/output)

5. CPU or MPU

6. Control, calculation and decision

7. Wires that carry information to and from sections of the computer

8. a. Data bus carries information.
   b. Address bus carries memory location or address.
   c. Control bus carries control signals to and from the CPU.

9. A temporary storage place in the CPU

10. A specialized register. Generally, all numbers moving into or out of the CPU must pass through the accumulator.

11. The IP always points to the next byte the microprocessor will use.

12. The ALU is the arithmetic logic unit; all math and logic operations are done in the ALU.

13. The fetch/execute cycle is a very precise sequence of steps.

    a. Fetch brings the op code number to the CPU.
    b. Execute performs the operation as instructed.

14. If a halt instruction is not given, the microprocessor will try to use the next number in memory as an instruction. Unpredictable results occur.

15. A binary (or hexadecimal) number that tells the CPU what type of operation to perform.

16. The number to be operated upon (moved, added, subtracted, etc.)

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT I: COMPUTER ARCHITECTURE

LESSON 3: 8088 ARCHITECTURE

I. Terms and definitions

A. Mnemonics--short, three- or four-letter words used for instructions to the MPU

B. Queue--method of lining up instructions within the CPU for faster access

C. Segmented address--method of dividing memory into segments of up to 64K each

D. Stack--segment of memory reserved by the MPU for storing and retrieving important information

II. Register groups

A. Temporary storage place for binary numbers within the MPU

B. 14 programmer-accessible registers in an 8088 divided into five functional groups (HO 3.1)

C. The segment group

1. The 8086/88 uses segmented memory addressing to quickly address any four 64K segments of total memory. The segment registers are all 16-bit registers.

2. Code segment register (CS) controls fetching instructions from computer memory.

3. Data segment register (DX) controls the memory segment where most memory read and write operations occur.

4. Stack segment register (SS) determines where stack operations occur.

5. The extra segment register (ES) is available if needed to help with another segment register. (example: a second data segment)

D. The general-purpose group

1. The four 16-bit word registers are data registers AX, BX, CX and DX.

2. The general-purpose registers can be used for several purposes.

a. Hold numbers

b. Read into them from memory

c. Perform calculations on the numbers

d. Write the register contents back into memory

3. Each of the general-purpose registers can be used as one 16-bit word storage register or as two independent 8-bit byte registers.

4. The general-purpose registers are considered as four 16- bit and eight 8-bit registers.

5. The 16-bit register AX can be used as AH (high byte) or AL (low byte) registers.

   a. Likewise, BX can be used as BH and BL.

   b. CX can be used as CH and CL.

   c. DX can be used as DH and DL.

6. Although the general-purpose registers are generally interchangeable, they each have a specialty.

   a. AX is the accumulator. Some operations have a shorter, faster op code that must use the AX register.

   b. Register BX is the base register.

   c. Register CX is the count register. Some forms of counting can be done automatically with CX.

   d. Register DX is the data register. DX is also used to hold the most significant word when performing 32-bit operations.

E. The index group

1. Index registers SI, DI, BP and SP specialize in pointing or indexing.

   a. Register SI is the source index.

   b. DI is the destination index register.

   c. BP is the base pointer.

   d. SP is the stack pointer.

2. When one of these 16-bit registers holds an address number, it is said to point to that address.

F. The flag group

1. A 9-bit register holds the flag group.

2. Each bit of the register is an independent flag that signals the status or condition of something that has happened or will happen.

3. Flags are sometimes called the condition code register.

III. Additional sections of the 8088

A. The instruction pointer, a 16-bit register, points to the next instruction to be fetched.

1. The IP is like the steering wheel guiding a car in the proper direction.

2. The IP can steer the 8088 straight ahead, backward or forward as needed.

B. The 8086/88 MPU architecture is divided into two major sections for faster and more efficient operation.

1. BIU (bus interface unit)

2. EU (execution unit)

C. The BIU (bus interface unit) handles the routine data bus and address operations.

1. Segment registers and IP are in the BIU.

2. With segmented addressing, the address segment (which segment?) is added to an offset (where within the segment?) held in the IP.

3. The address generator sums the two numbers to get an absolute 20-bit address.

4. The bus control logic determines which bus and what direction for the data to flow.

5. If incoming data is an op code or operand, it is passed to the execution unit through the queue.

6. The queue is a series of registers that hold several sets of numbers, pre-fetched from the bus, lined up ready for the execution unit.

7. Internal communications registers are used by the BIU and are not accessible by the operator.

D. The EU (execution unit) holds the index group of registers, flag group, general-purpose group and ALU.

1. When the EU control system needs another instruction, it takes one off of the queue rather than fetch it directly from memory, thus saving time.

2. The EU specializes in manipulating data and decoding and executing instructions.

3. Temporary registers are used by the EU and are not accessible by the operator.

IV. Instruction modes

A. Mnemonic

1. Mnemonics represent the binary op code numbers.

a. Computers understand binary numbers.

b. Words are easier for human operators to understand.

2. The mnemonic for move is MOV.

3. The mnemonic for addition is ADD.

4. The mnemonic for subtraction is SUB.

B. Register immediate instructions

1. An immediate instruction means that the value to be moved is part of the instruction. $_H$ indicates that the number is in hexadecimal; <u>don't</u> enter it.

Example: MOV AX,FFAA$_H$

2. After MOV, the destination is given first.

   a. In the example, the move is into register AX.

   b. The mnemonic MOV is the op code.

3. The value to be moved, $FFAA_H$, is given last. This value is the operand.

4. The immediate form is shown below.

Do what?	Where to?	With what?
MOV	AX	$FFAA_H$
MOV	destination,	immediate value

5. The immediate form holds the value in the instruction.

   a. The immediate is similar to LET A=3 in BASIC.

   b. The values are constants.

C. Other instruction modes

1. Move immediate value to memory.

   Example: MOV (memory location),$12AE_H$

   a. Move 12AEH to a memory location.

   b. The value is in the instruction.

2. Move register to register.

   Example: MOV AX,BX

   a. Contents of BX are moved to AX.

   b. Register contents can be changed.

   c. The values are variables.

   d. The value is held in the register, not in the instruction.

3. Move register to memory.

   Example: (memory address),AX

   a. Move contents of AX to a memory address.

   b. The value is held in the register, not in the instruction.

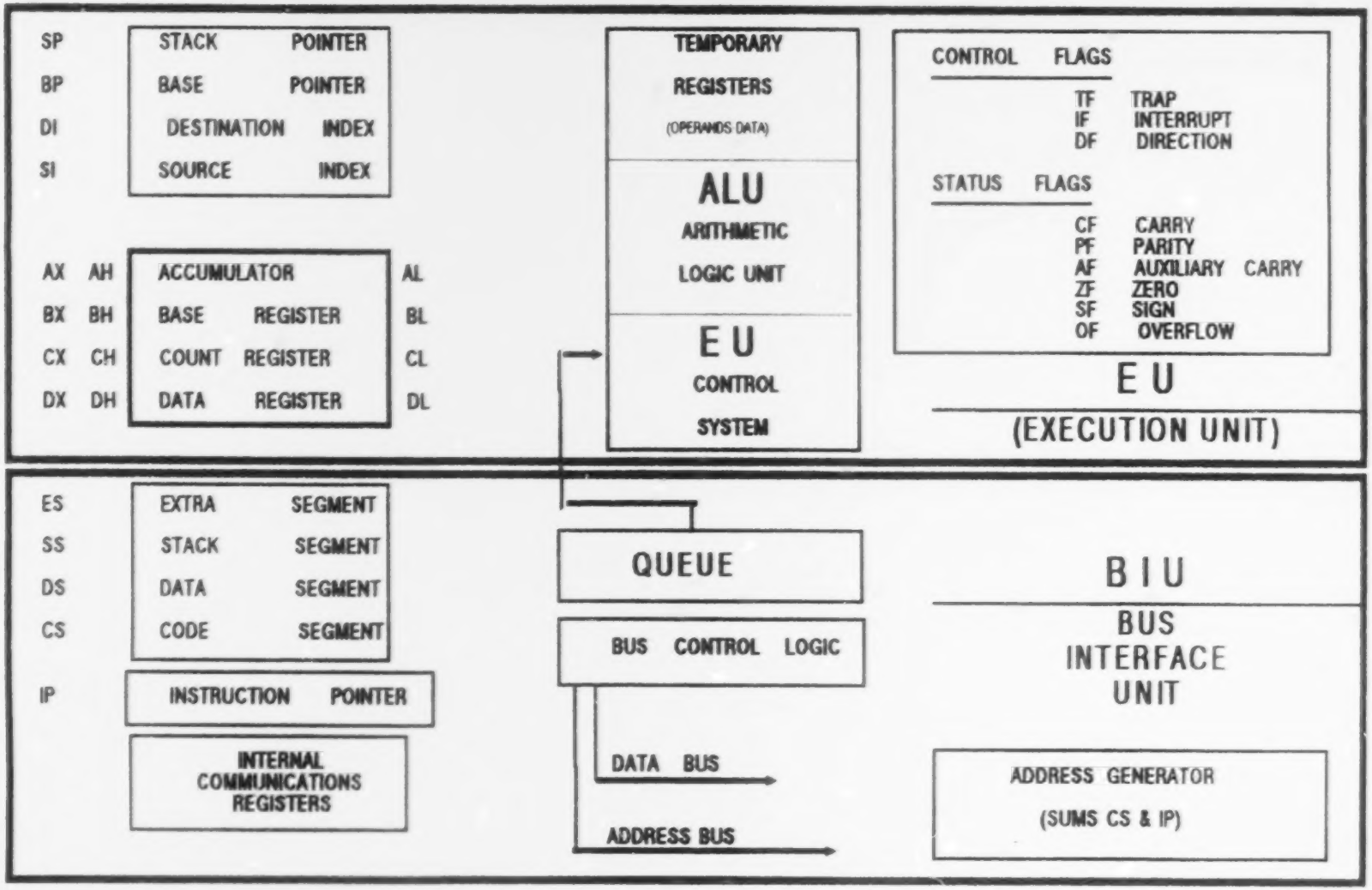
4. Move memory to register.

   Example: MOV AX,(memory location)

   a. Move contents of a memory location to AX.

   b. The value is held in memory, not in the instruction.

THE 8086 / 8088 CPU



DEBUG COMMANDS

Debug
Debut — Start Debug session

Q
Quit — End Debug session

H
Hex — Hex arithmetic

R
Register — Show registers

Rxx
Register xx — To show any register

RIP
Register IP — Change IP

RAX
Register AX — Change AX

E
Enter — Enter op codes in memory

A
Assemble — Enter mnemonics and assemble op codes

U
Unassemble — Display memory op codes

U200
Unassemble — List op codes starting at 0200

D
Dump — Display memory, ASCII

D200
Dump — Dumps starting at 0200

T
Trace — Single step program, display registers after each instruction

G
Go — Run program to end

G 0109
Go to 109 — Set breakpoint; run program to breakpoint

N
Name — Name of program to load or save

NFILE.COM
Name is FILE.COM

W
Write — Write to disk (save
Get byte count, low order in CS, high order in BX

To SAVE file

1. Name
2. Get byte count; load CX and BX
3. Write to disk

L
Load — Loads .COM file from disk

To LOAD file from inside Debug

1. **Name file**
2. Load

To load file from DOS

1. DEBUG FILE.COM

F
Fill — Fills block of memory with data

F100 200 AA
Fills block starting at 0100, ending at 0200 with hex AA

IBM ASCII codes

Dec.	Hex	Character
01	01	Happy face
02	02	Happy face
03	03	Heart
04	04	Diamond
05	05	Club
06	06	Spade
07	07	BEEP
08	08	Card
09	09	O
10	0A	Card
11	0B	Male
12	0C	Female
36	24	$

PRACTICE WITH DEBUG AND THE 8088

NOTE: Some commands used in this job sheet are: Debug, Q, H, R, RIP and U. Refer to HO 3.1, Debug Commands. Procedures used in this exercise are evaluated in a summary at the end of this job sheet.

Equipment:

DOS master diskette with Debug
IBM-compatible computer
Diskettes

Procedure 1: Hexadecimal addition and subtraction

1. Boot DOS. Load DEBUG.COM by typing Debug from the DOS command line. Debug will sign on with a - prompt.

2. From the Debug - prompt you can enter the Debug Hexadecimal calculator mode by typing H followed by two hex numbers. Type - H 5 3 (no comma--just separate the numbers with one space) and Debug will respond with:

   ```
   -H 5  3
   0008  0002
   ```

   a. The number on the left is the HEX SUM of 5H and 3H, and the number on the right is the HEX DIFFERENCE of 5H and 3H
   b. Debug will always give both the sum and difference of two hexadecimal numbers by using the H command.

3. Try adding and subtracting 2 and 1, 1A and FF, 8 and 2, E and 8, D and A. Calculate the answers yourself for practice and to verify Debug's answers.

4. Enter -H 1 2. What is the sum of 1 + 2 = ? _____. What is 1-2=? _____ What happened? _____

5. Calculate the two's complement of -1. _____ Convert answer to hex. _____ Is the same answer that Debug gave? ____

   a. The H command always gives the two's complement of a negative number.
   b. Debug's H command will be useful later when for adding or subtracting hex numbers.

Procedure 2: Examine and change instruction pointer

1. Only one register at a time can be viewed on the small microprocessor trainer boards.

   a. Debug will display ALL of the internal registers on the screen at one time.
   b. It is like looking inside of the silicon chip.

2. To display the 8086/88 registers, type R (for registers). You should now see a display similar to Figure 1.

3. The numbers on the screen might be different because of the amount of RAM memory in your machine.

   a. The four-general purpose registers, AX, BX, CX and DX are at the top of the display.
   b. Notice that those four registers all contain 0000_H.

```
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000 SI=0000
DI=0000
DS=37D8  ES=37D8  SS=37D8  CS=37D8  IP=0100   NV UP EI PL NZ
NA PO NC
37D8:0100 2020          AND     [BX+SI],AH          DS:0000=CD
```

Figure 1

4. The IP register is the instruction pointer (called program counter in some microprocessors).

   a. The IP always points to the next address that the microprocessor will go to for an instruction or data.
   b. Notice that Debug always starts with the IP set at 0100.
   c. To point to another address, IP will have to be changed.

5. Type RIP (R for register and then IP for which register to manipulate). Debug will respond with the current contents of IP.

   ```
   -RIP
   IP 0100 :
   ```

6. The colon (:) means that Debug is waiting for a new address

   a. Type 0200_H.
   b. Debug accepts the new number and shows the prompt -.
      :0200 -
   c. Type R to display the registers.

```
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000
DI=0000  DS=5F0D  ES=5F0D  SS=5F0D  CS=5F0D  IP=0200  NV UP EI
PL NZ NA PO NC 5F0D:0200 E8CD20      CALL      20CF
-
```

Figure 2

   d. The screen should be similar to Figure 2.
   e. IP now contains 0200_H.

7. Change IP to $FFFF_H$.

   a. Change IP to other hex numbers.
   b. Display the registers each time to verify that the contents of IP changed.

8. Change IP back to 0100_H and verify that the contents of IP are correct.

Procedure 3: Examine and change general-purpose registers

1. The contents of the general-purpose registers can be changed, also.

   a. Consider the registers to be similar to variables in BASIC.

b. Notice that the content of all four general-purpose registers is 0000_H.
c. Change the contents of register AX by typing RAX.
d. Debug will display the contents of AX.
-RAX
AX 0000 :
e. Type $FFFF_H$ to load register AX.
f. Type R to display the registers.
g. The display should be similar to Figure 3.

```
-R
AX=FFFF   BX=0000   CX=0000 DX=0000   SP=FFEE   BP=0000 SI=0000
DI=0000  DS=5F0D  ES=5F0D SS=5F0D  CS=5F0D  IP=0100 NV UP EI PL
NZ NA PO NC 5F0D:0100 7465      JZ      0167
```

Figure 3

h. Register AX now contains $FFFF_H$.
i. Type R for register, then the register name to access that register.

2. Deposit $AAAA_H$ in the BX register.

a. Deposit 1234_H in the CX register.
b. Deposit 5678_H in the DX register.
c. Display the registers.
d. The display should be similar to Figure 4.

```
-R
AX=FFFF  BX=AAAA  CX=1234  DX=5678  SP=FFEE  BP=0000  SI=0000
DI=0000 DS=5F0D  ES=5F0D  SS=5F0D  CS=5F0D  IP=0100   NV UP EI
PL NZ NA PO NC 5F0D:0100 7465      JZ      0167
```

Figure 4

3. Practice changing the contents of the AX, BX, CX, DX and IP registers and displaying the registers until you are comfortable in doing so.

4. To QUIT Debug type Q at the - prompt. Debug will end and the computer will return to DOS as indicated by the >A prompt

5. Restart Debug.

a. Examine the registers.
b. What are the contents of AX ____, BX ____, CX ____, DX ____, IP ____ ?
c. When Debug is started, the general-purpose registers always contain 0000_H and the IP always contains 0100_H.

Procedure 4: Examine and change memory address

1. Recall that the IP always points to the next memory address that the 8088 will go to for an instruction or for data.

a. When you display the registers, the current memory address is displayed at the lower left of the register display.

b. The lower middle of the display shows the op code (and operand) that corresponds to the hex number contained in the current memory location or address.

2. Since no program has been placed in this memory location, the hex numbers listed are simply "garbage" depending upon the state of each flip-flop in the memory location.

   a. Figure 5 shows
      37D8:0100 2020 AND [BX+SI],AH
   b. The screen display will probably show a number other than 37D8.

```
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000 SI=0000
DI=0000  DS=37D8  ES=37D8  SS=37D8  CS=37D8  IP=0100  NV UP EI
PL NZ NA PO NC 37D8:0100 2020       AND       [BX+SI],AH
DS:0000=CD
```

Figure 5

3. Remember that the 8088 microprocessor has a limit of four hexadecimal digits, or 16 binary digits.

   a. 2 raised to the 16th power binary digits imposes a limit of 65,535 (usually referred to as 64-K) memory locations or address's.
   b. We know that the 8088 can actually access over 1M of memory.
   c. This is done by adding two numbers together with one of the numbers offset by four binary digits.
   d. In effect, the sum is a 20-digit binary number.

4. The number on the left of the colon, 37D8 in Figure 5, is the number of a 64K segment.

   a. Each segment begins at a multiple of 16 bytes.
   b. The number on the right of the colon is the offset into the segment, or the byte in the segment.
   c. The actual address then is given in the form SEGMENT:OFFSET.
   d. The 37D8:0100 shown in Figure 5 then, is SEGMENT 37D8, OFFSET 0100.
   e. The segment address is stored in register CS, the code segment register.
   f. The offset address is stored in the IP register.
   g. Observe that the CS and IP registers do, in fact, contain the same address as displayed in the lower left of the register display.
   h. Debug always starts by setting the offset to 0100.
   i. For now, when we look at an address we will use only the offset; the amount of memory in your PC will determine the segment.

5. Set the IP to 0200_H.

   a. Display the registers
   b. The segment address is ____.
   c. Has the segment changed?
   d. The offset address is ____.
   e. Has the offset changed?
   f. Is the offset pointing to the same address is IP?
   g. Set the IP back to 0100_H.

6. Verify by displaying the registers.

Procedure 5: Display and change memory

1. The register display shows the contents of only one memory location, the address pointed to by CS:IP.

   a. Verify that the IP is set to 0100_H.
   b. To see the contents of a section of memory, starting at 0100_H (or where ever the IP points) type the U (Unassemble) command.

2. The display should be similar to Figure 6.

   a. The starting address CS:IP is xxxx:0100_H.
   b. The xxxx means whatever number your computer shows.

3. The display will be different because the contents of each memory location, or address, is random "garbage" from when you turned the computer on. Nothing has been stored in the memory locations yet.

```
-U
5DB7:0100 3025    XOR     [DI],AH
5DB7:0102 205468  AND     [SI+68],DL
5DB7:0105 7275    JB      017C
5DB7:0107 2E      CS:
5DB7:0108 205265  AND     [BP+SI+65],DL
5DB7:010B 61      DB      61
5DB7:010C 64      DB      64
5DB7:010D 2022    AND     [BP+SI],AH
5DB7:010F 000C    ADD     [SI],CL
5DB7:0111 46      INC     SI
5DB7:0112 69      DB      69
5DB7:0113 6C      DB      6C
5DB7:0114 65      DB      65
5DB7:0115 20666F  AND     [BP+6F],AH
5DB7:0118 756E    JNZ     0188
5DB7:011A 64      DB      64
5DB7:011B 3B20    CMP     SP,[BX+SI]
5DB7:011D 204573  AND     [DI+73],AL
-
```

Figure 6

4. Examine Figure 6. The starting address CS:IP is 5DB7:0100_H.

   a. Let's call the address 0100_H for now, ignoring the CS address.
   b. The next column to the right is 3025_H.
   c. Remember this is a random number, but Debug will show what instruction a number would represent if it were a real instruction.
   d. The display will always look like a real program, even if it is "garbage".
   e. The number 3025_H represents a two-byte instruction.
   f. The 30_H is one byte; the 25_H is the second byte.
   g. The next column shows the op code or instruction represented by 3025_H, XOR, followed by the operand (or operands) for that instruction. [DI],AH

5. Since the instruction starting at 0100_H was a two-byte instruction, it used memory address 0100_H and 0101_H.

   a. The next address displayed, then, is 0102_H.
   b. The instruction starting at 0102_H is a three-byte instruction, 20 54 68_H.
   c. The op code and operand is AND [SI+68],DL.
   d. Since this was a three-byte instruction, memory locations 0102_H, 0103_H and 0104_H are used and displayed on the same line.

6. The next instruction starting at address 0105_H is a two-byte instruction. The instruction starting at address 0107_H is a one-byte instruction, so the following instruction starts at address 0108_H.

7. Give the Un-assemble command again.

   a. Notice that the memory display started at the next offset address and after the ending offset address from the previous display.
   b. Type U again.
   c. Again, the display continues from the previous address because the offset address pointer IP still holds the next address to go to.

8. The starting address can be changed by giving the Un-assemble command followed by the desired offset starting address.

   a. Type U 0500_H.
   b. The IP is now set to address 0500_H and the display starts at that address.

9. Both segment and offset can be selected in this way.

   a. Type U $0000:0600_H$.
   b. The segment address (CS) is set to 0000_H and the offset address (IP) is set to 0600_H.
   c. This is only 600_H bytes from the very bottom of RAM!
   d. This is the area on the PC memory map where MS DOS loads into memory when the computer is first started.

10. Quit Debug.

    a. Restart Debug.
    b. Display the registers.
    c. What is the CS:IP address? ____ : ____

11. When Debug starts, the registers and flags are set to the following parameters.

    a. Debug always starts with IP pointing to 0100_H.
    b. This is the beginning address of the user RAM area on the PC memory map.
    c. The segment registers (CS, DS, ES and SS) are set to the bottom of free memory.
    d. With Debug loaded into memory, the bottom of free memory is the first segment after the top of Debug.
    e. The stack pointer (SP) is set to the end of the segment or the bottom of the transient portion of the program loader, whichever is lower.
    f. Flags are set to their cleared values.
    g. The remaining registers (AX, BX, CX, DX, BP, SI and DI) are set to zero.

12. If Debug has been started with a filespec, (a file loaded) the CX register contains the length of the file in bytes.

13. If the file length is greater than 64K, the length is contained in registers BX (the high byte) and CX (the low byte).

14. Quit Debug.

15. Return diskettes to storage and turn computer power off.

Procedure:	Evaluation 3	2	1	N
1. Hexadecimal addition and subtraction	—	—	—	—
2. Examine and change IP	—	—	—	—
3. Examine and change general-purpose registers	—	—	—	—
4. Examine and change memory address	—	—	—	—
5. Display and change memory	—	—	—	—
Average of above evaluations*	—	—	—	—

*This average is the student's final evaluation for competency I1: Identify and describe computer architecture.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT I: COMPUTER ARCHITECTURE

UNIT TEST

Identify true statements with an "a" and false statements with a "b."

____ 1. Memory that holds information, even after power is turned off, is called ROM.

____ 2. The Apple II computer uses an 8088 family microprocessor.

____ 3. The Amiga computer uses a 68000 family microprocessor.

____ 4. The IP or program counter steers the microprocessor to the next instruction to execute.

____ 5. Flags are sometimes called condition code registers.

____ 6. Debug is used to view programs written in BASIC.

____ 7. All addition or subtraction is done in the IP.

____ 8. The data bus carries information to and from the MPU.

____ 9. ROM is a non-volatile memory.

____ 10. The contents of a memory location is its address.

____ 11. The microprocessor was invented by the Apple computer company.

____ 12. The Commodore 64 uses the 6502 family of microprocessor.

____ 13. Instructions are brought into the MPU during the fetch cycle.

____ 14. An ADD instruction causes two numbers to be added together during the execute cycle.

____ 15. The first microprocessor was the Intel 4-bit 4004.

____ 16. The 8088 has 16-bit internal registers.

Select the option which answers the question or completes the statement.

____ 17. Where does the MOV DX,1A3X$_H$ instruction hold a value?

a. Memory
b. Register
c. Instruction
d. ALU
e. Operand

____ 18. What is a specific location in memory called?

a. Mnemonic
b. Op code
c. Operand
d. Address
e. Register

___ 19. What are the three main functions of a CPU?

a. Control, calculation, decision
b. CPU, memory, I/O
c. Data, address, control
d. MPU, CPU, ALU
e. Op code, mnemonic, operand

___ 20. Which is a three-letter word that represents instructions to the CPU?

a. Mnemonic
b. Op code
c. Operand
d. Address
e. Register

___ 21. What is made up of an ALU, registers and decoding circuit?

a. Microprocessor
b. Microcomputer
c. Peripheral
d. Interface
e. Mainframe

___ 22. What is a number that is to be operated upon called?

a. Mnemonic
b. Op code
c. Operand
d. Address
e. Register

___ 23. Which microprocessor family uses segmented addressing?

a. Z-80
b. 6502
c. 6800
d. 8088
e. 68000

___ 24. Which describes general-purpose registers in the 8088 microprocessor?

a. Four 16-bit registers
b. Eight 8-bit registers
c. Sixteen 4-bit registers
d. Both a and b
e. Both b and c

___ 25. Where is the value held in the instruction, MOV BL,DL?

a. Memory
b. Register
c. Instruction
d. ALU
e. Op code

___ 26. Which describes a device such as a disk drive or printer?

a. Microprocessor
b. Microcomputer
c. Peripheral
d. Interface
e. Mainframe

___ 27. What are the three types of busses?

a. Control, calculation, decision
b. CPU, memory, I/O
c. Data, address, control
d. MPU, CPU, ALU
e. None of the above

___ 28. In general, what component do all numbers to be processed pass through?

a. Register
b. Accumulator
c. Counter
d. Bus control logic
e. Instruction decoder

___ 29. Which is the point where two units connect within a system?

a. Microprocessor
b. Microcomputer
c. Peripheral
d. Interface
e. Mainframe

___ 30. Which section of the computer connects the CPU to the outside world?

a. MPU
b. IP
c. I/O
d. ALU
e. All of the above

___ 31. What is a binary or hexadecimal microprocessor instruction called?

a. Mnemonic
b. Op code
c. Operand
d. Address
e. None of the above

___ 32. What are the three sections of a typical computer?

a. Control, calculation, decision
b. CPU, memory, I/O
c. Data, address, control
d. MPU, CPU, ALU
e. None of the above

___ 33. Which of the following describes a register?

a. Temporary storage place in the MPU
b. Temporary storage place in memory
c. Permanent storage place in the MPU
d. Permanent storage place in memory
e. Memory address

___ 34. What is a system containing a MPU, memory and I/O called?

a. Microprocessor
b. Microcomputer
c. Peripheral
d. Interface
e. Register

___ 35. Where is a value held for the memory location instruction, MOV BX?

a. Memory
b. Register
c. Instruction
d. ALU
e. Decoder

___ 36. Where is the stack located?

a. Accumulator
b. MPU
c. IP
d. Memory
e. ALU

___ 37. Which microprocessor listed below has both a 16-bit bus and 16-bit internal registers?

a. 6502
b. 6800
c. 8086
d. 8088
e. None of the above

___ 38. Which is the brain of a computer?

a. IP
b. I/O
c. Bus
d. CPU
e. Memory

___ 39. What does an op code tell?

a. BASIC where to find a string
b. The microprocessor what to do
c. The assembler what to do
d. The operator of the computer's progress
e. None of the above

ANSWERS TO UNIT TEST (I1)

1. a
2. b
3. a
4. a
5. a
6. b
7. b
8. a
9. a
10. b
11. b
12. a
13. a
14. a
15. a
16. a
17. e
18. d
19. a
20. a
21. a
22. c
23. d
24. d
25. b
26. c
27. c
28. b
29. d
30. c
31. b
32. b
33. a
34. a
35. a
36. d
37. c
38. d
39. b

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT II: OPERATING SYSTEMS

UNIT OBJECTIVE

Upon completion of this unit, the student should be able to describe purposes of operating systems and use DOS commands. This knowledge will be evidenced by achieving the specified score on the job sheet and unit test.

SPECIFIC OBJECTIVES

After completion of the lessons in this unit, the student should be able to:

1.1 Define terms.

1.2 Explain the purpose of a disk operating system.

1.3 Discuss types of operating systems.

1.4 Describe how data is recorded on floppy disk drives.

1.5 Explain differences in DOS versions.

2.1 Explain uses of DOS.

2.2 Explain the function of common DOS commands.

2.3 Identify other disk-handling procedures.

2.4 Use common DOS commands. (competency I3, JS 1.1)

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT II: OPERATING SYSTEMS

CONTENTS OF THIS UNIT

A. Objective sheet

B. References

C. Notes to the instructor

D. Lesson plans

1. Lesson 1: THE DISK OPERATING SYSTEM (DOS)

   a. Information outline

   b. Assignment sheet

      AS 1.1: DOS terminology

   c. Answers to assignment sheet

2. Lesson 2: USING DOS COMMANDS

   a. Information outline

   b. Job sheet

      JS 2.1: Use common DOS commands

E. Unit test

F. Answers to unit test

REFERENCES

A. Books

1. Brey, Barry B. 8086/8088 Microprocessor: Architecture, Programming, and Interfacing. Columbus, OH: Merrill, 1987.

2. Brey, Barry B. The Z80 Microprocessor: Hardware, Software Programming and Interfacing. Englewood Cliffs, NJ: Prentice-Hall, 1988.

3. Coffron, James W. Microprocessor Programming, Troubleshooting, and Interfacing: The Z80, 8080, and 8085. Englewood Cliffs, NJ: Prentice Hall, 1988.

4. Dailey, Denton J. Small Computer Theory and Applications. New York: McGraw-Hill, 1988.

5. Gaonker, Ramesh. The Z80 Microprocessor: Architecture, Interfacing, Programming, and Design. Columbus, OH: Merrill, 1988.

6. Gillmore, Charles M. Microprocessors: Principles and Applications. New York: McGraw-Hill, 1989.

7. Hall, Douglas V. Microprocessors and Interfacing: Programming and Hardware. New York: McGraw-Hill, 1986.

8. Harman, Thomas L. The Motorola MC68020 and 68030 Microprocessors: Assembly Language, Interfacing, and Design. Englewood Cliffs, NJ: Prentice-Hall, 1989.

9. Hughes, Fredrick W. Microprocessor Technology: Theory and Experimentation. Englewood Cliffs, NJ: Prentice-Hall, 1989.

10. Kersey, Roger M. Personal Computer Operation and Troubleshooting. Englewood Cliffs, NJ: Prentice Hall, 1989.

11. Lafore, Robert. Assembly Language Primer for the IBM PC & XT. New York: The Waite Group, 1984.

12. Miller, Michael A. The 68000 Microprocessor: Architecture, Programming, and Applications. Columbus, OH: Merrill, 1988.

13. Norton, Peter. Inside the IBM PC. New York: Brady, 1986.

14. O'Connor, Patrick Joseph. Digital and Microprocessor Technology. 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1989.

15. Pasahow, Edward J. Microprocessor Technology and Microcomputers. New York: McGraw-Hill, 1988.

16. Tocci, Ronald J., and Lester P. Laskowski. Microprocessors and Microcomputers: Hardware and Software. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1987.

B. Curriculum guides

1. Brown, A. O. Advanced Microcomputer Service Technician. Stillwater, OK: Mid-America Vocational Curriculum Consortium, 1985.

NOTES TO THE INSTRUCTOR

A. Suggested approaches to teaching this unit

1. Show students how to warm boot and cold boot a computer.

2. Demonstrate the use of DOS commands.

3. Demonstrate proper handling of diskettes.

B. Suggested student activities

1. Practice formatting diskettes and copying files.

2. Practice using the PC and DOS commands.

C. Suggested supplemental material

1. Microcomputer Fundamentals (video, E-87-350). Sedalia, MO: The Media Center, State Fair Community College, 1987. (Available for purchase from SFCC; available for loan from Missouri Vocational Resource Center, phone 800/392-7217 or 314/882-9610)

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT II: OPERATING SYSTEMS

LESSON 1: THE DISK OPERATING SYSTEM (DOS)

INSTRUCTOR'S NOTE: These definitions may be specific to IBM or IBM-compatible machines.

I. Terms and definitions

A. ASCII--American National Standard Code for Information Interchange

B. Boot--process of looking for and loading into memory specific binary programs from the diskette. Programs are used by the microprocessor to read the keyboard, operate the disk drives, printer and video display.

C. Cold boot--process computer goes through when power is first applied to the PC. All POST tests are performed before booting the DOS.

D. Default drive--the disk drive that DOS uses unless told otherwise. The > prompt shows the default drive letter.

E. Hidden files--programs or files on the diskette that do not show up on the directory. The file names are hidden from view. Hidden files are not copiable with the DOS COPY command.

F. POST (Pre-Operation Self Tests)--power-on self-test; done by IBM and compatible computers when power is first turned on

G. Warm boot--process used to restart the microprocessor without turning off power; bypasses the POST commands. Warm boot is initiated from the keyboard by pressing Control, Alternate and Delete (CTRL-ALT-DEL) keys at the same time.

II. DOS introduction

A. DOS (disk operating system)--program on a level between the BIOS and the user or user's application programs

1. The user can use the powerful commands built into DOS to perform tasks easily.

2. Rather than write them again, application programs also use the operating system routines already written.

B. PC power up

1. The computer's microprocessor only understands instructions in binary number form.

2. A set of Basic Input Output System (BIOS) instructions are stored in read-only memory (ROM) on the computer system's circuit board.

3. When the computer is turned on, the microprocessor "wakes up", looking for a few very basic instructions and at specific locations or addresses in ROM BIOS.

4. Start-up instructions tell the microprocessor to start a series of self-tests called POST.

5. Each memory location is tested to determine the type of video board, type of display, and number and type of disk drives.

6. The keyboard is sent a message to test itself and report back.

7. Input/output (I/O) ports are checked, looking for peripherals (printer, mouse, joy stick, modem).

8. The status report from the keyboard is received.

9. If everything tests OK, a red light on drive A will light briefly and the loudspeaker will sound one short beep.

10. Any other signals mean a problem has been found.

    a. An error code will display on CRT screen.

    b. The computer will refuse to operate.

C. Boot up

1. The microprocessor starts disk drive A.

2. It then looks for additional more advanced instructions on the diskette.

3. If a hard disk is installed and floppy drive A is empty, the microprocessor looks for the advanced instructions on the hard drive.

4. Specifically, the microprocessor looks for the following.

    a. A file called COMMAND.COM

    b. Two "hidden" files

    c. A part of the disk operating system

5. The instructions contained in hidden files and COMMAND.COM are loaded into the computer's memory.

6. Next, the computer next looks for a batch file named AUTOEXEC.BAT to execute it.

    a. Can set clock/calender board if installed

    b. Set PATH if using a hard disk

    c. Automatically start a program

    d. Any other instructions

7. The computer then prints the DOS prompt on the screen. This lets the operator know the computer is ready for additional instructions.

    a. A:> (if using floppy diskettes)

    b. C:> (if using a hard drive)

8. If an IBM PC does not have floppy drives or a DOS diskette in drive A, the PC will come up in cassette BASIC. (Cassette BASIC does not recognize floppy drives.)

9. Non-IBM machines will give an error code if no floppy drives or DOS diskettes are found.

D. Hardware/software

1. Hardware is the physical parts and circuits of a computer.

2. Software, the binary instructions on diskette or tape, can be loaded into computer memory.

3. Firmware is the software imbedded in silicon ROM I.C.s.

III. DOS

A. There are several operating systems available.

1. CP/M80 was used on older 8-bit machines using the Z80 or the 8080 microprocessors.

2. Apple DOS was used with the Apple II.

3. TRSDOS is common on older Radio Shack Z80 machines.

4. DOS is available for many computers, including IBM, Xenix, Unix, CP/M86, Concurrent DOS and others.

B. Only two types of operating systems are most widely used on IBM PCs and compatibles.

1. MS-DOS

a. The "generic" version

b. 16-bit operating system

c. For the 8088/8086 microprocessor

d. Produced and sold by Microsoft

e. For use in compatible or clone computers

2. PC-DOS

a. Slightly modified form of MS-DOS

b. 16-bit operating system

c. for the 8088/8086 microprocessor

d. Sold by IBM

3. Either version will run on an IBM PC, PS/2 or compatible.

4. The two types of DOS commands are internal and external.

5. A copy of the DOS internal commands are read from the DOS diskette and placed in the computer's memory.

6. Internal commands can be called from memory even if the DOS diskette is removed from the computer.

7. External commands are on the diskette, which must be in the computer when the command is called.

   Example: FORMAT

IV. Bits, bytes and nibbles

A. Information is stored in computer memory as binary numbers.

1. A bit is a single binary digit–1 for high and 0 for low.

2. A byte is usually a group of 8 binary bits.

   Example: 10011011

3. Binary bits are difficult for humans to count and use.

   a. The hexadecimal numbering system is easier to use.

   b. Four bits represent one hex digit.

   Examples: 0010 binary = 2 hex = 2 decimal
   1010 binary = A hex = 10 decimal
   1111 binary = F hex = 15 decimal

4. Four bits is called a nibble.

5. One byte, then, equals two nibbles, two hex digits or eight binary digits.

6. The computer understands only numbers, which can represent any of the following.

   a. Decimal number

   b. Letter of the alphabet

   c. Control signal in the ASCII code

   d. Instruction or opcode

   e. Memory location

7. It depends upon how the computer interprets the binary pattern of the byte.

8. Since the days of the 8-bit microprocessors, the 8-bit data format has been used.

   a. Most available hardware

   b. The ASCII code

   c. The 8088 microprocessor data

   d. Diskette data

V. Versions of DOS

A. 5 1/4" floppy disk drives

1. These drives are used in the IBM PC and compatibles.

   a. Record 40 circular magnetic tracks on the disk

   b. Each track broken up into sectors

2. These drives can be floppy diskettes or a hard disk composition.

   a. Circular or disk-shaped magnetic recording material

   b. Material very similar to audio or video cassette tape

3. A new audio or video cassette can be inserted into the recorder and used immediately to record or play back information.

4. A computer disk must be pre-recorded or formatted before it can be used.

   CAUTION: Make sure you know how to issue the FORMAT command properly. An expensive program diskette (or hard disk) can be erased with an improper FORMAT command.

B. First issue of DOS: Version 1

   1. Version 1 recorded eight sectors on each track.

   2. Each sector contains 512 bytes of information.

   3. There are 40 tracks on a diskette side.

   4. A single-sided diskette, then, holds 512 bytes on eight sectors through 40 tracks--163,840 bytes of information.

   5. Each side of a double-sided diskette holds 512 bytes on eight sectors though 40 tracks--327,680 bytes.

C. Information recorded with nine sectors on each track since Version 2

   1. A single-sided diskette, then, holds 512 bytes times nine sectors times 40 tracks--184,320 bytes (180K).

   2. A double-sided diskette holds twice as much, or 368,640 bytes (360K).

   3. Many of the popular 3 1/2" diskettes have 80 tracks each side.

      a. Will hold 730,112 bytes (720K)

      b. Sometimes up to 1.2 megabytes (million bytes)

D. Newer versions

   1. Each new version of DOS uses the nine-sector format.

   2. Though improved with additional features, the nine-sector format takes more memory and diskette space.

   CAUTION: Don't mix programs with various DOS levels.

E. Filenames

1. Filenames on DOS diskettes can be up to eight letters long.

2. A period after these eight letters allows an additional three digits and is called an **extender**.

   Examples: TURBO.EXE REPORT.DOC COMMAND.COM (pronounced "command-dot-com")

   a. Executable binary files have .COM or .EXE extenders.

   b. It is common to use .DOC for document or text files.

3. While not required, using extenders is a good practice because they help classify types of files on a disk.

F. Diskette care

1. A write-protect notch is present on the side of each diskette.

   a. When it's uncovered, the disk can be read from, written to or formatted.

   b. When covered, the disk can be read from but not written to or formatted.

2. Take several steps to prevent diskette damage.

   a. Never fold or crease a diskette.

   b. Never touch the magnetic material with fingers.

   c. Never write on diskettes with pens or pencils.

   d. Keep diskettes in room temperature when in use; allow cold disks to warm up before use.

   e. Never expose diskettes to the magnetic field present in monitors (especially color), plug-in disk drives, magnetic-tipped screwdrivers or telephones.

G. Key points to remember

1. POST tests the computer system upon boot up.

2. A disk operating system must be "loaded" into the computer's memory before the computer can be used.

3. A floppy diskette must be pre-recorded or formatted before it can be used in the computer.

4. FORMAT can erase programs on a disk if used improperly.

DOS TERMINOLOGY

1. How many letters can a filename have in the name and in the extender?

2. When the computer first "boots", what files does it look for?

3. How many sectors does DOS Version 1 record on a disk track?

4. What does the write-protect notch do?

5. What does the term DOS mean?

6. What is the heart of a microcomputer?

7. How do you warm boot the computer?

8. How many tracks are on one side of a floppy diskette?

9. How many sectors are in a track for DOS version 2.1?

10. How many bits are in a byte? _____ A nibble? _____

11. What does BIOS mean?

12. What is the difference between PC-DOS and MS-DOS?

13. What are three guidelines for handling disks?

14. How is the file AUTOEXEC.BAT pronounced?

ANSWERS TO AS 1.1

1. Up to eight letters in the filename, and up to three in the extender
2. The computer looks for COMMAND.COM and two hidden files.
3. DOS Version 2 has nine sectors on a disk track.
4. When covered, the disk is write-protected; when uncovered, the disk can be written on.
5. Disk operating system
6. The microprocessor
7. Press CTRL, ALT and DEL keys at the same time.
8. 40 tracks on one side
9. Nine sectors
10. Eight bits in a byte, four bits in a nibble
11. BIOS stands for Basic Input Output System.
12. PC-DOS is marketed by and specifically for IBM. MS-DOS is the generic version.
13. Any three of the following:
    Never fold or crease a diskette
    Never touch magnetic material with fingers
    Never write on diskette with pen or pencil
    Never expose diskette to a magnetic field
14. Autoexec-dot-bat

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT II: OPERATING SYSTEMS

LESSON 2: USING DOS COMMANDS

NOTE: Upon completion of this lesson, the student should be able to start the PC, format a diskette and use common DOS commands.

I. DOS uses

- A. Start up or boot the computer
- B. Set time and date
- C. Get a diskette ready to be used (FORMAT)
- D. Copy a diskette (DISKCOPY)
- E. Compare two diskettes (DISKCOMP)
- F. Copy a file (COPY)
- G. Compare two files (COMP)
- H. Find out what files are on a diskette (DIR)
- I. Look at an ASCII file (TYPE and)
- J. List the path and contents of each directory (TREE)
- K. Change a file's name (RENAME)
- L. Remove a file from a diskette (ERASE or DEL)
- M. Check the quality of disk files (CHKDSK)
- N. Copy and verify files and directories (XCOPY)
- O. Clear the screen (CLS)
- P. Read DOS version number (VER)
- Q. Read disk volume name (VOL)

II. Common DOS commands

- A. DATE and TIME
  1. Sets the internal clock and calender manually
  2. Special clock/calender board available for automatic setup

B. FORMAT

1. Use the FORMAT command to get a diskette ready to receive information.

   a. Records tracks and sectors

   b. Checks the diskette for bad spots

   c. Builds a directory to hold information about files

2. Use FORMAT only once per diskette--when it is new.

   CAUTION: If a diskette is formatted that has files already on it, those files will be erased!

3. A "bootable" disk contains two hidden files and COMMAND.COM.

4. Start the computer in one of the following manners.

   a. With a DOS disk, then change to the program diskette

   b. With the program disk if it has hidden files and COMMAND.COM

5. When using a hard disk, DOS and the program are already on the same drive.

6. Switches can used with the FORMAT command.

   a. /s transfers system files to new disk

   b. /v allows naming of the volume or diskette

   c. /1 formats single-sided only

C. DISKCOPY (best used for unformatted disks only)

1. Use the DISKCOPY command to make a backup copy of an entire diskette onto another diskette.

   a. Formats disk as it copies

   b. Covers up anything already on the target diskette

   c. Copies everything, including any bad spots

2. This command is not used to copy one file; use the COPY command for only one file.

3. The source diskette is the original to be copied from.

4. The target diskette is the new diskette being copied to.

NOTE: It is a good habit to back up program diskettes as soon as they are purchased or created. Put the original away in a safe place and use the copy.

D. DISKCOMP

1. This command compares the backup diskette to the original.

2. Use DISKCOMP to make sure the backup diskette was copied correctly.

   a. Indicates if two diskettes are identical

   b. Indicates differences between diskettes

3. DISKCOMP cannot compare a floppy diskette to a hard disk.

E. COPY

1. Used to copy one or more files from one diskette to another or to copy all files from one diskette to another

2. Formatted target diskette needed

3. Does not erase files already on target diskette

4. Can copy a file and change the name of the new file

F. COMP

1. Use COMP command to compare file or files.

2. Make sure that backup file is identical to source (original) file.

3. DOS will prompt you for the primary file name of the file (on source diskette) and the secondary or copied file name.

G. DIR

1. Use the DIR command to find out what files are on a diskette.

2. The DIR command displays various information.

   a. All the files on the default drive

   b. The volume label (if any)

   c. The name of the directory being listed

   d. The files in that directory

3. The screen display will show the filename.

   NOTE: The period between filename and extension does not show on the CRT screen but must be used when typing the complete filename.

   a. The extension

   b. The size of the file (in bytes)

   c. The date and the time that file was written

   d. The amount of free space left on the diskette (in bytes)

4. One line is displayed for each file on the diskette.

5. Pause the screen when necessary.

   a. Use /P option (/P is a switch).

   b. The display will stop when the screen is full.

   c. Hit any key to scroll the next screen full.

6. For many files, use the wide display option switch.

   a. Switch /W

   b. Gives a wide display of filenames only

H. TYPE

1. This command lets the operator look into a file without retrieving it.

2. TYPE displays the contents of a file on the screen.

3. ASCII text files are displayed in a legible format.

4. Binary program files may not be readable.

   a. Characters are op codes and operands, not alphabetic or numeric codes.

   b. Binary files are displayed as if they were ASCII codes.

   c. The display is not readable.

5. Pause screen keys can be used.

   a. This is helpful if the information scrolls off the screen too quickly to read it.

   b. Press both CTRL and NUM-LOCK keys at same time.

6. Printer Echo keys produce a hard copy of the screen display.

   a. Press both Shift and PRTSC keys at same time

   b. To stop echoing to the printer, press the keys again.

I. TREE: to list the tree or path of each directory, subdirectory or file to the screen

J. RENAME: changes a file's name, its extension or both

K. ERASE or DEL

1. Use the ERASE command to delete files from diskettes.

2. Removing old files makes room for new files on a diskette.

IMPORTANT: When using ERASE, plan ahead--check typing before pressing the Enter key. Once a file has been erased, you cannot get it back!

L. CHKDSK

1. Use CHKDSK to verify the following.

   a. Diskette sectors

   b. Tracks

   c. File allocation tables

   d. For fragmented files

   e. For lost clusters

2. Diskette and system memory statistics are displayed.

3. If files are fragmented or clusters of sectors lost, use the /F option.

   Example: CHKDSK/F

   NOTE: CHKDSK/F should be used occasionally, especially on a hard disk, to keep the disk and file allocation tables "healthy".

M. XCOPY

1. Use XCOPY to copy files and directories to a formatted diskette.

2. XCOPY differs from COPY.

   a. XCOPY copies files into memory, then to a diskette.

   b. COPY copies files directly from diskette to diskette.

3. XCOPY verifys that the copy is identical to the original file.

   a. Add the /V switch to the XCOPY command.

   b. The copy process will take longer but will be more reliable.

N. CLS

1. Clears the screen

2. Moves the cursor to the upper left corner of the screen

O. VER: to read the version of DOS being used

P. VOL: to read the disk volume name, if present

III. Additional information

A. Copy-protect the diskette

1. Copy-protect tabs placed over the diskette's notch prevents accidental erasure.

2. It is a good idea to protect DOS or system diskettes and original program diskettes.

B. Good practice

1. It is generally best to FORMAT a diskette.

2. Then use COPY or XCOPY to copy files to diskette rather than DISKCOPY.

3. FORMAT will find bad spots on the diskette

4. DISKCOPY, on the other hand, will not find bad spots and will copy bad spots over to the new disk.

5. The newer 3 1/2" diskettes are much less susceptible to error than the 5 1/4" diskettes because of their rigid construction.

USE COMMON DOS COMMANDS

NOTE: This job sheet is evaluated by the checklist on the last page. After satisfactorily performing this job sheet, ask your instructor if you may keep a copy of it for future reference.

Equipment:

IBM PC or compatible
DOS diskette
Scratch diskette

Starting or loading DOS

1. Remove diskette from its paper envelope. Do not touch the magnetic material in the diskette.

   NOTE: The newer 3 1/2" diskettes are much less susceptible to error than the 5 1/4" diskettes because of their rigid construction.

2. Open the drive door.

   NOTE: Flipping the door latch open may break it. Hold the latch with your fingers and gently open the latch. On half-height drives, turn the latch lever to the horizontal position.

3. Slide the diskette into the drive with the label side up.

4. Make sure the diskette is all the way in, and shut the drive door. On half-height drives, turn the lever down to lock. Don't jam the diskette.

If the computer is off

1. Switch on the printer, the video monitor and then the computer.

2. The computer will test itself. The length of the pause will depend on how much memory and what is connected to the computer--the more memory, the longer the pause.

3. The computer will beep once, the diskette drive light will come on while DOS is being read, and you will hear some clicks and whirs. This procedure is know as a cold boot.

If the computer is on

1. Press and hold the system reset keys:
   Ctrl + Alt + Del then release them all.

2. The diskette drive light will come on while DOS is being read, and you will hear some clicks and whirs. This procedure is known as a warm boot.

DATE & TIME

1. When DOS is loaded and ready, you will see a message similar to this on the screen:
   Current date is Tue 1-01-1980
   Enter new date:__

2. The __ is the cursor and shows where the first number typed will appear.

Telling DOS the date

1. Enter one or two numbers (between 1 and 12) for the month.

2. Enter a dash (-) or a slash (/).

3. Enter one or two numbers (between 1 and 31) for the day.

4. Enter another dash or slash.

5. Enter the last two numbers of the year (between 80 and 99).

6. Press the Enter key. (Enter is the large key on the right of the keyboard with a left pointing arrow.)

7. The Enter key places the command into memory for the computer to read. It is also called the Return key.

Telling DOS the time

1. After entering the date, DOS displays a message similar to this:

```
Current time is 0:01:05.58
Enter new time:
```

The time displayed is in hours, minutes, seconds and hundredths of seconds.

2. Enter one or two numbers (between 0 and 23) for the hours. Note that 24-hour time is used instead of 12-hour time.

3. Enter a colon (:). A dash or slash will not work.

4. Enter one or two numbers for the minutes (between 0 and 59). To enter the seconds and hundredths of a second, refer to the TIME command in the DOS reference book.

5. Press the Enter key. DOS enters the time and sets the remaining values to zero automatically.

NOTE: It is not necessary to re-enter date and time; simply press the enter key to respond to the prompt.

When DOS is ready

1. DOS will now sign on with version number, copyright information and A>.

2. The A> is the DOS prompt. It signals that DOS is ready and it is ready for instructions.

Giving DOS a command

1. Wait for the DOS prompt, A>.

2. Enter the command and any other parts the command requires.

3. Commands can be entered in upper-case or lower-case letters or a combination of both.

4. Use a blank (press the Space bar) to separate the parts of the command from each other.

5. Press the Enter key when finished.

NOTE: The DOS command must be the first thing on the input line.

Specifying the default drive

1. The A in the prompt designates the default drive.

2. DOS searches the diskette in the default drive for any commands or filenames that are entered unless the operator specifies another drive in the file specification.

3. To change the default drive:

```
A>      (original prompt)
A>  B:      (new drive designation)
B>      (new prompt)
```

4. Drive B is now the default drive.

   NOTE: Use the colon (:) after the drive letter. The system will now search drive B for commands and file names.

Formatting a disk

CAUTION: If you FORMAT a diskette that has files already on it, the files will be erased!

Format a "bootable" disk

1. Insert the DOS diskette in drive A.

2. To make a "bootable" or system disk, type:
   FORMAT B:/V/S (or FORMAT A:/V/S if using one floppy drive)
   (This formats drive B, lets you name the volume or diskette (switch /V), transfers two hidden system files needed (switch /S) and transfers COMMAND.COM to the new formatted diskette.)

3. This message appears:
   Insert new diskette for drive B: and strike any key when ready

   NOTE: If you have two drives, insert the new diskette in drive B, if it is not already there. Removing the DOS diskette is unnecessary.

4. Now press the Enter key.

5. When the format is complete, DOS will tell you that system files have been transferred and ask for a volume name.

6. Use your name, the program name or any other identification name.

7. From now on, the DIR command will also display the volume name of the diskette.

Format a diskette for files only with a volume name

1. To use the diskette for files only with a volume name, type:
   FORMAT B:/V (or FORMAT A:/V if using one floppy drive)

2. This message appears:
   Insert new diskette for drive B: and strike any key when ready

   NOTE: With two drives, insert the new diskette in drive B, if it is not already there. It is not necessary to remove the DOS diskette.

3. Now press the Enter key. When the format is complete, DOS will tell you that system files have been transferred and ask for a volume name.

4. Use your name, the program name or any other identification name.

5. From now on, using the DIR command will also display the volume name of the diskette.

   CAUTION: When the disk is inside, never mark on the diskette sleeve with a ballpoint pen. Use felt tip marker only. The best way is to mark on a label and then stick the label to the diskette. Handle diskettes carefully--no folding, scratching, marking or touching with fingers. Keep them in the paper sleeve when not in the computer.

DISKCOPY

1. Use the DISKCOPY command to make a backup copy of an entire diskette on another diskette.

   NOTE: It is a good habit to back up program diskettes as soon as they are purchased or created. Put the original away in a safe place and use the copy.

2. Make sure DOS is ready and A:> is displayed.

3. Insert the DOS diskette in drive A, if it is not already there.

4. Key in this: DISKCOPY A: B:

5. Press the Enter key.

6. This tells DOS to copy everything from drive A over to drive B and cover up anything that is on drive B.

7. If you have two drives, this message appears:
   Insert source diskette in drive A:
   Insert target diskette in drive B:
   Strike any key when ready

8. Now remove the DOS diskette.

9. Insert the source diskette in drive A and formatted target diskette in drive B, if you have not already done so.

10. It is not necessary to switch diskettes.

11. If you have one drive, this message appears:
    Insert source diskette in drive A:
    Strike any key when ready

12. Remove the DOS diskette in drive A, if you have not already done so. (If you are copying the DOS diskette, leave it in the drive.)

13. Insert the source diskette in drive A.

14. Now press any key.

    NOTE: Depending on the amount of memory in your computer, you may have to switch diskettes.

15. The "In use" light comes on while the source diskette is being read, and then this is displayed:
    Insert target diskette in drive A:
    Strike any key when ready

16. Remove the source diskette.

17. Insert the target diskette.

18. Now press any key.

19. The "In use" light comes on while the backup diskette is being written until this message appears:
    Copy complete
    Copy another (Y/N)?

20. Press N (for no).

DISKCOMP

Comparing the backup diskette

1. Make sure the backup diskette that was just made using DISKCOPY was copied correctly.
2. Use the procedures practiced for single-drive or two-drive computers.
3. Key in this: `DISKCOMP A: B:`
4. If the diskettes are not identical, copy and compare again.
5. Note that DISKCOMP cannot be used to compare a floppy diskette to a hard disk--only one floppy to another floppy disk.

COPY

1. Use the COPY command for either of the following options.
   a. Copy one or more files from one diskette to another.
   b. Copy all files from one diskette to another formatted diskette.

Copy one or more files

1. Make sure DOS is ready and A:> is displayed.
2. Insert the diskette with the file on it (the source diskette) in drive A.
3. Key in the file's name like this: `COPY A:filename B:`
4. Key in the full name of the file you wish to copy.
5. Notice where the spaces are located in the command.
6. Press the Enter key.
7. If you have two drives, follow the messages to insert diskettes.
8. If you have one drive, this message appears:
   ```
   Insert diskette for drive B: and strike any key when ready
   ```
9. Remove the source diskette.
10. Insert the destination or backup diskette.
11. Now press any key.
12. When the copy has been made, this message is displayed:
    ```
    1 File(s) copied A:>
    ```

Copy file and change file name

1. COPY can be used to copy a file and change the name of the new file in the same step.
2. For example, to copy drive A's file named SAMPLE.TXT to a new file named TEXT.DOC on drive B, use the following command:
   ```
   COPY A:SAMPLE.TXT B:TEXT.DOC
   ```
3. The file could be copied to the same disk but with a different name.
   ```
   COPY A:SAMPLE.TXT A:TEXT.DOC
   ```

Copy all files

1. For an example of the COPY command, the next steps will copy all files from the DOS diskette to your formatted system diskette.

2. Use the diskette that you used the /S option on.

3. If you have one drive, DOS will treat it as both drive A and B.

4. You must switch diskettes each time DOS tells you to exchange diskettes until you see the message:
   X Files copied (where X is the number of files copied) A:>

5. Insert the DOS diskette in drive A. (This is the source diskette.)

6. If you have two drives, insert the formatted diskette in drive B. (This is the destination or target diskette.)

7. Type the COPY command appropriate for your situation.
   COPY A:*.A: (if using one floppy drive) --Or--
   COPY A:*.B: (if using two floppy drives)

8. This instructs DOS to copy all files on the diskette in the source drive to the diskette in the destination drive. The (*) is a global command meaning to copy ALL filenames with ALL extenders.

9. Now press the Enter key.

10. DOS will copy all the files. When the A:> appears, copying is completed.

COMP

1. Use COMP command to make sure the file or files that you just copied to the backup diskette using COPY is identical to the source (original) diskette or file.

2. Make sure DOS is ready and A:> is displayed. Insert the DOS diskette in drive A.

3. Key in this: COMP

4. Press the Enter key.

   NOTE: If you get the following message, place the original (source) diskette in the drive and press any key.
   Insert diskette for drive A: and strike any key when ready

5. DOS will prompt you for the primary file (the name of the file on the source diskette). This message appears: Key in primary file name

6. When you see the message, enter: A:filename

7. Key in the file's name from the source diskette for filename and press the Enter key.

8. Then this message appears: Key in 2nd file name or drive ID

9. Key in the full name of the file you copied to the destination diskette. Press enter.

10. The names of the files being compared are displayed.

11. If you have one drive, this message appears:
    Insert diskette for drive B: and strike any key when ready
    Remove the source diskette from drive A. Insert the target diskette in drive A. Press any key.

12. When files have been compared and the files are identical, this message appears:
    Files compare ok
    Compare more files (Y/N)?

13. Key in Y or N.

    NOTE: On some computers you might get the message:
    Insert COMMAND.COM disk in drive x: and strike any key when ready
    In response, insert the DOS diskette in drive A and press any key. Then A:> will be displayed.

14. If the files are not identical, go through the copying and comparing steps again.

DIR

1. Use the DIR command to find out what files are on a diskette.

2. Key in DIR. The screen display will show:
   a. The filenames
   b. The extension
   c. The size of the file (in bytes)
   d. The date and the time that file was written
   e. The amount of free space left on the diskette (in bytes)

3. One line is displayed for each file on the diskette.

4. The DIR B: command option will display the directory of drive B. Key in DIR B:

Screen pause option switch
1. Key in DIR/P

2. The screen will display the disk directory.

3. The display will stop when the screen is full. Hit any key to scroll the next screen full.

Wide screen display switch
1. Key in DIR/W.

2. This option will give a wide display of filenames only. (Date, time and bytes are not listed.)

TYPE

1. The TYPE command lets you "look into" a file.

2. TYPE displays the contents of a file on the screen.

Text files
1. Look for an ASCII file on the directory with a .BAT or .DOC extender.

2. Key in: TYPE filename

3. Note that ASCII text files are displayed in a legible format.

Non-text files
1. Key in TYPE COMMAND.COM (or any other .COM or .EXE file)

2. Note that binary files do not display a readable text. Only ASCII text files can be typed.

Pause screen

1. If at any time information scrolls off the screen too fast to be read, use the following pause screen keys:
   Ctrl Num Lock (Press both at same time)

Printer echo

1. Use the printer echo keys to get a hard (paper) copy of what appears on the screen.
   Shift PrtSc (Press both at same time)

2. To stop echoing to the printer, press the keys again.

TREE

1. To list the path of each directory and subdirectory and files on drive C to the screen, use the TREE command as follows: TREE C: /F | MORE --or-- TREE C: /F

2. List the path without using the | MORE option.

3. To send the tree to the printer, use the command: TREE C: /F >PRN

RENAME

1. The RENAME command lets you change a file's name or its extension.

2. Key in: RENAME old filename new -filename

3. For example, to rename SAMPLE.TXT to HELP.DOC, key in:
   RENAME A:SAMPLE.TXT HELP.DOC

4. Notice that the current name is first, then a space, and then the new name. Examine the directory; you will find the filename changed.

ERASE

1. Use the ERASE command to remove files that are no longer needed from diskettes.

2. The form is: ERASE filename or DEL filename

3. DOS assumes that you mean the file is on the default drive. Always tell DOS which drive to use.
   ERASE B:filename or ERASE A:filename

4. Check the disk directory to see that the file has been erased from the disk.

   IMPORTANT: Plan ahead and check your typing before pressing the Enter key when using ERASE. Once a file has been erased, it cannot be un-erased from DOS.

CHKDSK

1. Use CHKDSK to check the diskette sectors, tracks and file allocation tables. Place the DOS diskette in drive A and key in:
   CHKDSK

2. Observe that the diskette and system memory statistics displayed. If files are fragmented or clusters of segments lost, use the /F option.
   CHKDSK/F

   NOTE: CHKDSK/F should be used occasionally, especially on a hard disk to keep the disk and file allocation tables "healthy".

XCOPY

1. To copy files and directories to a formatted diskette use the XCOPY command.

2. To copy from drive A to drive B use the following command:
   XCOPY A: B: /S /E /W

3. To copy from drive B to drive A use the following command:
   XCOPY B: A: /S /E /W

4. To copy from drive A to drive C use the following command:
   XCOPY A: C: /S /E /W

Verify switch

1. To have XCOPY verify that the copy is identical to the original file, add the /V switch to the command.
   Example: XCOPY A: C: /S /E /W /V

2. The copy will take longer but will be more reliable.

CLS

1. To clear the screen at any time enter CLS from DOS.

2. The screen will clear; the cursor and prompt will move to the top left corner of the CRT.

VER

1. To read the DOS version number, enter VER.

VOL

1. To read the disk volume name (if present), enter VOL.

2. Return diskettes to storage and turn computer power off.

Procedure:		Evaluation 3	2	1	N
1.	Starting or loading DOS	—	—	—	—
2.	Date and time	—	—	—	—
3.	FORMAT	—	—	—	—
4.	DISKCOPY	—	—	—	—
5.	DISKCOMP	—	—	—	—
6.	COPY	—	—	—	—
7.	COMP	—	—	—	—
8.	DIR	—	—	—	—
9.	TYPE	—	—	—	—
10.	TREE	—	—	—	—

11.	RENAME	—	—	—	—
12.	ERASE	—	—	—	—
13.	CHKDSK	—	—	—	—
14.	XCOPY	—	—	—	—
15.	CLS	—	—	—	—
16.	VER	—	—	—	—
17.	VOL	—	—	—	—
Average of above evaluations*		—	—	—	—

*This average is the student's final evaluation for competency I3: Describe the purpose of operating systems.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT II: OPERATING SYSTEMS

UNIT TEST

Mark true statements with an "a" and false ones with a "b".

____ 1. DOS means Disk Operating System.

____ 2. All the physical parts of a computer are called software.

____ 3. Touching the magnetic part of the floppy disk with the fingers can damage it.

____ 4. When the PC receives a cold boot, memory is tested, keyboard and option boards are tested, and if everything is OK, DOS is loaded.

____ 5. To record on a 5 1/4" disk, the write-protect notch must be covered.

____ 6. The command DISKCOPY A: B: will copy one file on the disk from drive A to drive B.

____ 7. Disks are divided into sides, tracks and sectors.

____ 8. The file COMMAND.COM must be on the boot-up disk.

____ 9. If A> were showing on the screen and B:<Enter> were typed, the DEFAULT drive would now be drive A.

____ 10. If memory, keyboard and option boards test OK but drive A is defective, an IBM-PC will boot up in cassette BASIC.

____ 11. A filename can have up to eight letters and up to eight extenders in the name.

____ 12. When the computer "boots", DOS looks for a file named AUTOEXEC.BAT; if it is found, DOS runs the batch file.

____ 13. A DOS command can have a /s or /1 or /v. The /x is called a switch.

____ 14. The command DISKCOPY A: B: will copy all files, including bad spots on the disk, from drive A to drive B.

____ 15. "Warm-booting" the computer by pressing CTRL-INS-DEL keys.

____ 16. When the PC receives a warm boot, memory, keyboard and option boards are tested; if everything is OK, DOS is loaded.

____ 17. The operating system works between the user and the computer to handle some of the routine instructions to the CPU.

____ 18. Physical parts and chips of a computer are called hardware.

____ 19. The operating system is written in English since the computer only understands words.

____ 20. PC-DOS and MS-DOS are the only disk operating systems for the IBM PC.

Match the term on the right with its definition on the left.

____	21.	Gives a full 80-column screen display of just the filenames	a. Bits
____	22.	Four bits	b. Boot
____	23.	Clears the screen and puts the cursor in the home (upper left) position	c. Byte
			d. CLS
____	24.	Shows complete directory with number of bytes in each file, time and date file was recorded	e. COPY
____	25.	Shows version number of DOS	ab. DEL
____	26.	Binary digit of 1 or 0	ac. DIR
____	27.	Start-up of computer	ad. DIR/W
____	28.	Erase a file	ae. Nibble
			bc. VER

Select the option which answers the question or completes the statement.

____ 29. How many letters can a filename and extender have?

a. Six in the name, six in the extender
b. Three in the name, three in the extender
c. Six in the name, three in the extender
d. Eight in the name, three in the extender
e. Three in the name, eight in the extender

____ 30. What microprocessor does the IBM-PC have?

a. 8080
b. 8085
c. 6800
d. 8088
e. 68000

____ 31. Upon successful completion of POST, what sound(s) does the speaker make?

a. A beep
b. Two beeps
c. One short and one long beep
d. One long and short beep
e. None of the above

____ 32. What files must a disk have on it to be a "bootable" disk?

a. Two "hidden" files and COMMAND.COM
b. COMMAND.COM, DATE.COM, TIME.COM
c. FORMAT.COM, COPY.COM, COMMAND.COM
d. COMMAND.COM, DATE.COM, TIME.COM, COPY.COM
e. None of the above

___ 33. When the computer first "boots", what three files does it look for?

a. Two "hidden" files and COMMAND.COM
b. COMMAND.COM, DATE.COM, TIME.COM
c. FORMAT.COM, COPY.COM, COMMAND.COM
d. COMMAND.COM, DATE.COM, TIME.COM, COPY.COM
e. None of the above

___ 34. Which operating system was written by Microsoft but is marketed by IBM to run on IBMs or compatible PCs?

a. IBM DOS
b. MS-DOS
c. PC-DOS
d. CP/M
e. Soft DOS

___ 35. A PC is on, the Ctrl-Alt-Del keys are pressed, memory is cleared, no tests are performed, and DOS loads and is ready to use. What is this process known as?

a. Format
b. DIR
c. Cold boot
d. Warm boot
e. XCOPY

___ 36. What is meant by the default drive?

a. Defective drive
b. Wrong drive
c. Drive indicated by the prompt
d. Master drive
e. None of the above

___ 37. Which command will copy one file from drive B to drive A?

a. FORMAT A: B:
b. DISKCOPY B: A:
c. COPY B:FILENAME A:
d. DISKCOMP B: A:
e. None of the above

___ 38. Which is true of the CLS command?

a. Only clears the screen
b. Closes files
c. Clears the screen and moves the cursor to upper left corner the of screen
d. Shuts down computer
e. None of the above

___ 39. A PC is turned on, checks memory, tests boards, loads DOS and is ready to use. What is this process known as?

a. Format
b. DIR
c. Cold boot
d. Warm boot
e. None of the above

___ 40. What is the generic operating system that was written by Microsoft that will run on IBM or any compatible PC?

a. IBM DOS
b. MS DOS
c. PC DOS
d. CP/M
e. Soft DOS

___ 41. When DOS version 2.1 is used, how many tracks and sectors are recorded on a diskette when formatting?

a. 39 tracks, eight sectors
b. 40 tracks, eight sectors
c. 39 tracks, nine sectors
d. 40 tracks, nine sectors
e. None of the above

___ 42. A PC is turned on, checks memory, tests boards, tests keyboard and drives. What is this process known as?

a. Operating system
b. DIR
c. Advanced diagnostics
d. POST
e. Format

___ 43. Which computer listed below does <u>not</u> use MS-DOS?

a. IBM PC
b. Apple IIe
c. IBM compatible
d. IBM PS/2
e. None of the above

___ 44. Which size of disk has a rigid construction that is less susceptible to handling error?

a. 2 3/4"
b. 3 1/2"
c. 4 1/2"
d. 5 1/4"
e. 8"

ANSWERS TO UNIT TEST (I3)

1. a
2. b
3. a
4. a
5. b
6. b
7. a
8. a
9. b
10. a
11. b
12. a
13. a
14. a
15. b
16. b
17. a
18. a
19. b
20. b
21. ad
22. ae
23. d
24. ac
25. bc
26. a
27. b
28. ab
29. d
30. d
31. a
32. a
33. a
34. c
35. d
36. c
37. c
38. c
39. c
40. b
41. d
42. d
43. e
44. b

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT III: COMPUTER PROGRAMMING

UNIT OBJECTIVE

Upon completion of this unit, the student should be able to identify and describe levels of computer languages. The student will write and debug simple BASIC language programs. This knowledge will be evidenced by achieving the specified score on the job sheets and unit test.

SPECIFIC OBJECTIVES

After completion of the lessons in this unit the student should be able to:

1.1 Identify terms and definitions.

1.2 Identify and describe low-level languages.

1.3 Identify and describe high-level languages. (competency I2)

2.1 Define programming terms.

2.2 Describe math procedures for programming.

2.3 Use BASIC language commands.

2.4 Write simple basic language programs. (competency I5, JS 2.1-2.4)

2.5 Debug simple basic language programs. (competency I6, JS 2.5)

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT III: COMPUTER PROGRAMMING

CONTENTS OF THIS UNIT

A. Objective sheet

B. References

C. Notes to the instructor

D. Lesson plans

1. Lesson 1: LOW-LEVEL AND HIGH-LEVEL LANGUAGES

   a. Inforrnation outline

   b. Assignment sheet

      AS 1.1: Languages

   c. Answers to assignment sheet

2. Lesson 2: BASIC LANGUAGE

   a. Information outline

   b. Job sheets

      JS 2.1: Write program in BASIC--Pt. 1
      JS 2.2: Write program in BASIC--Pt. 2
      JS 2.3: Write program in BASIC--Pt. 3
      JS 2.4: Write program in BASIC--Pt. 4
      JS 2.5: Debug software

E. Unit test

F. Answers to unit test

REFERENCES

A. Books

1. Power C Manual. Richardson, TX: Mix Software, 1988.

2. The Visible Computer: 8088. Bryan, TX: Software Masters, 1985.

3. Turbo Pascal Reference Guide. Scotts Valley, CA: Borland International, 1988.

4. Brey, Barry B. 8086/8088 Microprocessor: Architecture, Programming, and Interfacing. Columbus, OH: Merrill, 1987.

5. Dailey, Denton J. Small Computer Theory and Applications. New York: McGraw-Hill, 1988.

6. Gillmore, Charles M. Microprocessors: Principles and Applications. New York: McGraw-Hill, 1989.

7. Hall, Douglas V. Microprocessors and Interfacing: Programming and Hardware. New York: McGraw-Hill, 1986.

8. Harman, Thomas L. The Motorola MC68020 and 68030 Microprocessors: Assembly Language, Interfacing, and Design. Englewood Cliffs, NJ: Prentice-Hall, 1989.

9. Lafore, Robert. Assembly Language Primer for the IBM PC & XT. New York: The Waite Group, 1984.

10. Norton, Peter. Inside the IBM PC. New York: Brady, 1986.

B. Curriculum guide

1. Brown, A. O. Advanced Microcomputer Service Technician. Stillwater, OK: Mid-America Vocational Curriculum Consortium, 1985.

NOTES TO THE INSTRUCTOR

A. Suggested approaches to teaching this unit

1. Schedule time for each student on an IBM PC.

B. Suggested student activities

1. Practice writing programs given in the job sheets.

2. Debug by finding and correcting errors.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT III: COMPUTER PROGRAMMING

LESSON 1: LOW-LEVEL AND HIGH-LEVEL LANGUAGES

I. Terms and definitions

A. High-level language--written with English-like statements that are easy for humans to understand but must be converted to numbers for the microprocessor to understand

B. Low-level language--uses simple instructions understood by the microprocessor; can be written in binary or hexadecimal number system or simple instructions assembled into binary or hexadecimal numbers

C. Object code--binary numbers that the CPU understands. Source code is converted to object code by assembler, interpreter or compiler.

D. Reserved words--words used in a language as specific instructions or identifiers; may not be used for other purposes

E. Source code--instructions written by the human operator; can be numbers, mnemonics or English statements

F. Syntax--the "grammar" of a computer language, proper use of symbols, punctuation and reserved words

II. Low-level languages

A. Machine language

1. Machine language is a set of binary numbers that the microprocessor understands.

2. Each instruction has a number or op code.

3. Binary numbers are difficult for humans to work with.

a. Numbers can be written in hexadecimal form.

b. Hex is easier for humans to work with.

c. A monitor or executive program is on the computer which converts back and forth from hex to binary.

4. Instructions from all languages must eventually be converted to binary numbers.

5. Programs can be written on paper using a symbolic language.

a. Symbolic language uses three- and four-letter mnemonics, such as MOV for move a number and HLT for halt or stop.

b. Mnemonics are easier for humans to understand.

c. Computers do not understand mnemonics.

6. The programmer must hand-assemble the program code by reading the mnemonic, looking up its hex number and entering the hex number into the computer.

7. The advantages of machine language follow.

   a. Runs fastest of any language--10 times faster than BASIC

   b. Small program binary code--only one-tenth the space of BASIC

   c. Takes less space in memory

   d. Takes less space on disk

   e. Can directly access registers and ports

   f. Can manipulate the CPU without the aid of BIOS and DOS

8. There are also disadvantages to consider.

   a. A machine language program takes 10 times longer to write than BASIC.

   b. The programmer must know the architecture of the specific microprocessor.

   c. Each microprocessor family has different codes.

   d. Machine language is more difficult to learn than high-level languages.

B. Assembly language

1. Assembly is a symbolic language written with text editor.

2. Mnemonics are used for source code instructions.

3. The source code is passed through an assembler, which looks up binary numbers for op codes and converts source code to machine language.

4. Assembly language's advantages follow.

   a. Runs 10 times faster than BASIC

   b. Less space for program object binary code--only one-tenth the space of BASIC

   c. Takes less space in memory

   d. Takes less space on disk

   e. Can directly access registers and ports

   f. Can manipulate the CPU without the aid of the BIOS and DOS

   g. Easier for programmer than machine language because labels or names are used for addresses and subroutines

5. There are disadvantages, as well.

   a. The program takes longer to write--10 times longer than BASIC.

   b. The programmer must know the architecture of a specific microprocessor.

c. Each microprocessor family has different codes.

d. It is more difficult to learn than high-level languages.

III. High-level languages

A. General information

1. Source code statements and instructions are written in English.

2. High-level languages must be converted to binary object code for MPU.

3. High-level source code is converted to binary object code by interpreter or compiler.

4. An interpreter reads the source code and converts to object code one line at a time, executing that set of instructions before reading the next line and executing it.

a. Interpreters interpret every time the program runs.

b. The interpreter language is loaded first.

c. The program cannot run alone.

d. The editor and debugger are part of the program.

5. A compiler reads the source code and converts the whole program to object code at one time.

a. The object code can then run alone.

b. A compiler program is faster than interpreted code.

c. A text editor and a debugger are needed.

B. BASIC

1. Relatively easy to learn

2. Supplied or available for most microprocessors

3. Available in interpreter or compiler versions

a. The most common is interpreter BASIC (i.e. IBM BASICA, GWBASIC and Applesoft BASIC).

b. BASIC compilers generate faster object code. Examples are Quick BASIC, Turbo BASIC and True BASIC.

C. PASCAL

1. A powerful general-purpose language

2. Highly structured language

a. Variables must be declared at beginning of program.

b. Subroutines called procedures are building blocks.

3. A compiled language that generates a fast, compact object code
    a. Microsoft PASCAL
    b. Turbo PASCAL

D. C language
1. A powerful general-purpose language
2. Highly structured language
    a. Variables must be declared at beginning of program.
    b. Subroutines called procedures are building blocks.
3. A compiled language that generates a fast, compact object code

    Examples: as Microsoft C, Turbo C and Power C
4. Has most of the programmer advantages of assembly language
5. Language of choice for many professional programmers
6. Highly portable
    a. Source code written on machine or microprocessor
    b. Can be compiled on another microprocessor

E. COBOL
1. Business/accounting language
2. Compiled language
3. Widely used on mini and mainframe computers

F. FORTRAN
1. Scientific/mathematics language
2. Compiled language
3. Widely used on mini and mainframe computers

LANGUAGES

Define the following terms.

1. Object code:

2. Source code:

3. Compiler:

4. Interpreter:

5. Assembler:

Provide a short answer for these questions.

6. Why is the hexadecimal number system used so much with computers?

7. List two low-level languages.

8. List four high-level languages.

9. List two advantages of a low-level language.

10. List two disadvantages of a low-level language.

11. List two advantages of a high-level language.

12. List two disadvantages of a high-level language.

ANSWERS TO AS 1.1

1. Binary numbers that the MPU understands. Source code is converted to object code by assembler, interpreter or compiler.

2. Instructions written by the human operator; can be numbers, mnemonics or English statements

3. Converts a high-level language source code to binary object code

4. Reads the high-level source code a line at a time and converts that line to object code

5. Converts low-level symbolic language to binary object code

6. The hexadecimal number system is easier for humans to understand than the binary system.

7. Machine and assembly

8. Any four of the following: BASIC, C, Pascal, COBOL, Fortran

9. Any two of the following:
   a. Runs faster than high-level language
   b. Object binary code is small
   c. Takes less space in memory
   d. Takes less space on disk
   e. Can directly access registers and ports
   f. Can manipulate the CPU without the aid of BIOS and DOS
   g. Is easier for programmer than machine language

10. Any two of the following:
    a. The program takes longer to write.
    b. Programmers must know specific microprocessor architecture.
    c. Each microprocessor family has different codes.
    d. It is more difficult to learn than high-level languages.

11. Any two of the following:
    a. Easier to learn
    b. Faster to learn
    c. Faster to write
    d. Do not have to know microprocessor architecture

12. Any two of the following:
    a. May be slower in execution
    b. Takes up more memory
    c. Takes up more disk space

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT III: COMPUTER PROGRAMMING

LESSON 2: BASIC LANGUAGE

NOTE: After this lesson, the student should be able to start the PC and write simple programs in BASICA or GWBASIC.

I. Terms and definitions

A. BASICA--advanced BASIC only on IBM PC computers. Part of the language is in ROM, part loaded into memory from disk.

B. Constant--a numeric value included in a program that does not change value

C. Floating point division--math operations using positive or negative numbers with fractional parts or only decimal fractions; can be large (498.7) or small values (0.00125)

D. GWBASIC--generic, advanced BASIC for IBM-compatibles. All of the language is on disk to be loaded into memory.

E. Integer division--math operations using only integers; value can be negative or positive with no fractional part

F. String--text of alphabetic characters or numbers without a numeric value

G. Variable--reserved place in memory that holds a numeric value. The memory location has a name, then values can be assigned to that name (LET x = 3.14). The value can change while the program is running.

Example: LET x = x + 1 would increment the variable x by 1 each time the program passed through the statement.

H. String variable--reserved place in memory that holds text characters (LET x $ = "TOM"). The $ indicates that the variable is a string. The string must be enclosed by quotes.

II. Math procedures

A. Mathematical symbols

1. Six symbols are used in BASIC programming.

a. + for addition

b. - for subtraction

c. * for multiplication

d. / for floating point division

e. \ for integer division

f. ^ for exponential numbers

2. Parentheses must be used to separate the various steps of a mathematical problem.

   Example:

   5 + 10 / 2 = 10
   (5 + 10) / 2 = 7.5

3. Example of algebraic formula $a = \frac{(2 + 4) \times 4}{3}$

4. Example of the same formula entered in the computer

   LET a = ((2 + 4) * 4) / 3

B. Hierarchy of mathematical symbols

1. Hierarchy determines which mathematics operation to do first.

2. Exponential numbers are calculated first.

3. Multiplication and division are next.

   a. Have the same precedence

   b. Executed as encountered from left to right

4. Addition and subtraction are done last.

   a. Have the same precedence

   b. Executed as encountered from left to right

C. Relational operators

Operator	Relation tested
=	Equality
< > or > <	Inequality
<	Less than
>	Greater than
< = or = <	Less than or equal to
> = or = >	Greater than or equal to

III. Commands and statements

A. Form

1. Command and its purpose

2. Format of command

   a. Brackets show what has been entered.

   b. Do not key in the brackets.

3. Comments or examples

B. Frequently used BASIC commands

1. AUTO
   a. Generates a line number each time Enter is pressed
   b. AUTO [number] [, [increment]]
   c. Default is 10,10
   d. Ctrl-C to exit
2. BEEP sounds the speaker
3. CLEAR
   a. Sets all numeric variables to zero
   b. Sets all string variables to null
   c. Clear [,numeric expression][,numeric expression]
   d. CLEAR
      (1) Clears all data from memory
      (2) Does not erase the original program
   e. CLEAR, 32768
      (1) Clears all data from memory
      (2) Sets maximum BASIC work space at 32K
4. CLS clears the screen to the background color
5. CONT (continue) resumes program execution after a break
6. DATA
   a. Stores the numeric and string constants that are accessed by the program's READ statement
   b. DATA constant [,constant]
   c. Can be numeric or string constant
   d. String constants surrounded by quotation marks if string contains commas, colons or significant leading or trailing blanks
   e. Can be placed anywhere in the program
   f. The READ statement accesses DATA statements in line number order
7. DELETE
   a. Eliminates entire program lines

b. DELETE [line1] [-line2]

8. DIM (dimension)

   a. Specifies maximum values for array variable subscripts and allocates storage accordingly

   b. DIM variable(subscripts) [,variable(subscripts)]

      Example: DIM Name$(24)

   c. Default is 10 if variable is not DIM'd.

9. EDIT

   a. Displays a program line for editing

   b. EDIT [line number]

10. END

   a. Terminates program execution; closes all files

   b. Use at end of program

11. FOR ... NEXT

   a. Performs a series of instructions in a loop

   b. Loops a given number of times

   c. Variable to be used as a counter

   d. x = the initial value of the counter

   e. y = the final value of the counter

   f. z = a number to be used as an increment; default of 1

```
FOR variable=x TO y [STEP z].
NEXT[variable][,variable]
```

12. GOSUB ... RETURN

   a. Branches to and returns from a subroutine

```
GOSUB [line number]
.
.
RETURN
```

13. GOTO

   a. Unconditional branch out of the normal program sequence or to a specified line number

   b. GOTO [line number]

14. IF

   a. Makes a decision regarding program flow

   b. Based on the result of an expression

   c. IF expression [,]THEN clause [ELSE clause]

d. IF expression[,]GOTO line[[,]ELSE clause]

e. Can be any numeric expression or a BASIC statement

f. Can also be a sequence of statements separated by colons

g. Clause sometimes the line number to branch to

h. Line is the line number in the program.

15. INPUT

a. To key in from keyboard during program execution

b. INPUT[;]["prompt";] variable [,variable]

c. When the computer encounters INPUT, it waits for input from the keyboard and displays a question mark on the screen. A comma after the prompt string suppresses a question mark.

d. Assigns data to numeric or string variable

e. Assigns data to variable after ENTER key is pressed

16. KEY

a. Sets or displays the soft keys

b. Display at bottom of CRT

c. KEY ON

d. KEY OFF

17. LET

a. Assigns value of an expression to a variable

b. [LET] variable=expression

Examples: LET A=99 A=99

c. Optional word

18. LIST

a. This command prints the program currently in memory.

b. It uses a screen or other specified device.

c. LIST [line1] [-[line2]] [,filespec]

d. LIST, lists the entire program on the screen.

e. LIST 35, lists line 35 on the screen.

f. LIST 10-20, lists lines 10 through 20 on screen

g. LIST 10-20,"LPT1:" lists lines 10 through 20 on the printer.

19. LLIST

a. This command lists all or part of program in memory on to the printer.

b. LLIST[line1][-line2]]

c. LLIST 10-20, lists lines 10 through 20 on the printer.

20. LOAD

a. A program is loaded from the specified device into memory and optionally run with this command.

b. LOAD filespec[,R]

c. LOAD "menu" loads menu but does not run it.

d. LOAD "menu",R loads and runs menu.

e. RUN "menu" also loads and runs.

f. LOAD"B:menu" loads menu from drive B.

21. LOCATE

a. Use this command to position the cursor on the active screen.

b. LOCATE [row],[col]

c. LOCATE down, across

d. Row is a numeric expression ranging from 1 to 25. It is the screen line number (how far down on the screen).

e. Col is a numeric expression ranging from 1 to 80. It describes the screen column number (how far across the screen).

f. LOCATE 1,1 moves cursor to upper left corner of screen.

g. LOCATE 12,40 moves cursor to center of screen, down 12 lines and across to column 40.

22. NEW: deletes the program in memory and clears all variables

23. ON..GOSUB
ON..GOTO

a. Conditional branch

b. Branches to specified line number

c. ON n GOTO line [,line]

d. ON A GOSUB 50 goes to line 50

e. ON A GOTO 80 goes to line 80

24. PRINT

    a. PRINT displays data on the screen.

    b. PRINT [list of expressions] [;]

    c. ? can be used instead of PRINT.

    d. The position of each printed item is determined by the punctuation used to separate the items in the list.

    e. BASIC divides the line into print zones; there are 14 spaces per zone.

    f. A comma causes the next value to be printed at the beginning of the next print zone.

    g. A semicolon causes the next value to be printed immediately after the last value.

25. PRINT USING

    a. Prints strings or numbers using a specified format

    b. PRINT USING "string expression";list of expressions[;]

26. RANDOMIZE

    a. Re-seed the random number generator

    b. RANDOMIZE [n]

    c. n = an integer expression used as the seed

27. READ

    a. Reads values from DATA statement

    b. Assigns value to variable.

    c. READ variable[,variable]...

    d. Example:
```
80 FOR I= 1 to 10
90 READ A
100 NEXT I
110 DATA 3,5,7,....
```

28. RENUM

    a. This command renumbers program lines.

    b. RENUM [newnum] [,[oldnum] [,increment]]

    c. Newnum is first line number to use in new sequence. The default is 10.

    d. Oldnum is line in program to begin. Its default is the first line of the program.

    e. Increment is the step to use. Its default is 10.

f. RENUM starts at 10 and steps by tens.

g. RENUM 300,,50 starts with 300 and steps by 50s.

29. RND

a. Returns a random number between 0 and 1

b. v = RND(x)

30. SAVE

a. Use this command to store a BASIC program file on diskette.

b. SAVE B:filename saves filename to drive B.

c. SAVE filename,A saves filename in ASCII format.

(1) Takes more space

(2) Required by some file merging

d. SAVE filename,P saves filename and PROTECTS it.

(1) Cannot be listed

(2) Cannot be edited

(3) Cannot be un-protected

31. TAB

a. This command spaces over to position n.

b. Tab can only be used with PRINT.

c. TAB(n) n must be 1 to 255.

d. TAB(25)

32. TRON
TROFF

a. Debugger to trace program execution

b. Used in indirect mode for debugging

c. TRON--tracer turned on

d. TROFF--tracer turned off

33. WIDTH

a. Sets width of screen display

b. WIDTH 80 or WIDTH 40

34. SOUND

a. This command generates sound through the speaker.

b. SOUND freq, duration

c. Freq is the desired frequency in Hertz.

(1) Lowest is 37

(2) Highest is 32767

d. Duration is the desired duration in clock ticks; each tick occurs 18.2 times each second.

e. SOUND 800,36 800 Hz for two seconds

f. SOUND ,0 to turn off

C. Text color information

1. COLOR sets the text colors.

2. COLOR [foreground] [,[background] [,border]]

3. In text mode, operator can set several variables.

a. Foreground--1 of 16 colors

b. Character blink, if desired

c. Background--1 of 8 colors

d. Border--1 of 16 colors

4. To define foreground or border colors, use numbers.

0	Black	6	Brown	11	Light cyan
1	Blue	7	White	12	Light red
2	Green	8	Gray	13	Light magenta
3	Cyan	9	Light blue	14	Yellow
4	Red	10	Light green	15	High-intensity white
5	Magenta				

5. Characters can be made to blink.

a. Set foreground equal to 16 plus number of desired color.

b. COLOR 20 sets red blinking characters.

6. Background colors are only colors 0-7.

7. COLOR 7,1,1

a. Foreground 7, white

b. Background 1, blue

c. Border 1, blue

8. COLOR 14,4,2

a. Foreground 14, yellow

b. Background 4, red

c. Border 2, green

WRITE PROGRAM IN BASIC--PT. 1

Equipment:

DOS and GWBASIC diskette (or BASICA)
IBM-compatible computer
File diskette

Procedure 1

Evaluation 3 2 1 N

1. Boot the PC. Load BASIC by typing GWBASIC. (Type BASICA if using IBM.) — — — —

   NOTE: The OK prompt is BASIC indicating that it's time to type.

2. Type in Listing #1. Press Enter at the end of each line. — — — —

(Listing #1)

```
10 REM ***  FIRST BASIC PROGRAM  ***
20 REM ***  TO PRINT YOUR NAME   ***
30 REM *** REM MEANS REMARK...ANYTHING AFTER REM IS IGNORED BY THE PC.
40 REM REMs ARE USED FOR MESSAGES TO THE OPERATOR.
50 REM PRINT YOUR OWN NAME WITHIN THE QUOTES WHEN YOU TYPE LINE #80.
60 REM NOTICE THAT WORDS, CALLED A STRING, THAT YOU WANT TO PRINT MUST
70 REM BE WITHIN QUOTATION MARKS.
80 PRINT "YOUR NAME"
90 REM LINE 80 TOLD THE COMPUTER TO PRINT WHATEVER IS WITHIN QUOTES
100 END :REM ALWAYS END YOUR PROGRAM WITH END!
```

3. Type RUN to execute the program. What did the program do? — — — —

4. Type LIST to list the program again. What lines actually make up the program? _____ Will the program run with just those lines? _____ — — — —

5. Prove your answer by erasing REM lines. Delete a line by simply typing the line number and ENTER. — — — —

   CAUTION: Don't type the number of a line that is to be kept.

6. After deleting unwanted lines, type LIST to show the remaining program. — — — —

7. Now RUN the program. Did it execute as expected? — — — —

8. Try adding lines to print other string variables. Don't forget to use line numbers. — — — —

9. Instead of the word PRINT, try using the ? and your string variable. Don't forget the quotes. — — — —

10. LIST the program (called a listing or source code). What happened to the ? when the program was listed? _ _ _ _

Procedure 2

1. Clear the program from the computer memory by typing NEW. Type in Listing #2. _ _ _ _

(Listing #2)

```
1 REM ***  PROGRAM TO DEMONSTRATE "STRING VARIABLES" AND INPUT COMMAND
2 REM THE DOLLAR SIGN TELLS BASIC THAT THE VARIABLE WILL BE MADE
3 REM OF LETTERS OR NUMBERS WITH NO NUMERIC VALUE.
4 REM
5 REM INPUT tells BASIC to wait for something
6 REM to be typed in from the keyboard.
7 REM The question mark ? will display to tell you BASIC is waiting.
8 REM
10 INPUT NM$
15 REM Wait to get string variable from keyboard
20 PRINT "Hello"
30 PRINT NM$
35 REM Then print it on the screen
40 END
```

2. RUN the program. When the question mark appears, type your name. Describe what happens. _ _ _ _

3. Notice that in order to print, a string must be within quotes. Since the string "Hello" will always print Hello, it is called a constant string. Notice that the string variable is not in quotes. NM$ is a variable because the contents can change or vary each time the program runs. _ _ _ _

4. Try it! RUN the program several times. Type in a different word or string each time. Does it work? _____ _ _ _ _

5. What is happening to the old text display each time the program is run? _ _ _ _

   Does it keep scrolling to the top of the screen? _____

6. Clear the screen of old garbage each time the program runs by typing the following: 8 CLS _ _ _ _

7. LIST the program. Notice than the new line #8 has been added to the listing at the proper place. RUN the program. What happened? _ _ _ _

   RUN it again. What happened?

   NOTE: Each time the program runs, the screen clears and starts a new display in the top left corner of the screen. Since we did not specifically tell it otherwise, the display location defaulted to the top left corner.

8. Now type RENUM. LIST the program. What happened? _ _ _ _

NOTE: RENUM renumbers the program lines starting with 10 and counts by 10. There is room between each line to add new lines if needed.

Procedure 3

1. What command clears the computer memory? _____ Clear the memory. _ _ _ _

2. Enter Listing #3. _ _ _ _

(Listing #3)

```
10 REM ***  Program to demonstrate the FOR-NEXT loop
20 REM
30 INPUT  NM$              :REM Get the name from the keyboard
40 FOR TIMES = 1 TO 5      :REM Start a counter, TIMES, to count to 5
50 PRINT NM$               :REM Print the string variable on the screen
60 NEXT TIMES              :REM Loop back to FOR
65 REM  Add 1 (increment by 1) to the counter and
66 REM  print the string variable again.
70 REM When counter has reached 5, FALL OUT OF THE LOOP
75 REM and go to the next program line.
80 PRINT "DONE"       :REM Then print DONE on the screen
90 END                :REM Always end your program with END
```

NOTE: Notice that REMs have been placed on the same line as the command by adding a colon (:) as a separator. This is the same as adding another physical line. As BASIC passes through the counter called TIMES (could be named anything), the counter is incremented by 1. The FOR must always have a NEXT paired with it to create a LOOP.

3. RUN the program. What happened? _ _ _ _

4. Now change the number that TIMES will count to by typing the following: EDIT 40 _ _ _ _

NOTE: This will bring line 40 down so that it can be changed or edited.

5. Use the arrow keys (on the keypad) and place the cursor just to the right of the number 5. Press the backspace key to delete 5. Replace the 5 with 10. _ _ _ _

6. When ENTER is pressed, the old line 40 will be replaced with the edited line 40. LIST the program. Is line 40 changed? _____ Explain what you expect the changed program to do. _ _ _ _

7. RUN the program. Was your prediction correct? _____ _ _ _ _

8. Try changing the count several times and observe the results. How could line 20 be changed so that it will clear the screen each time the program runs? _ _ _ _

9. Make the change and RUN the program. Did it work? ____ _ _ _ _

10. Delete the count number in line 40. Instead of a number, type in CNT. _ _ _ _

NOTE: Since there is no $, the variable CRT is a numeric variable. Each time BASIC comes to CNT, it will go to a place in memory called CNT and get the contents for the number to use.

11. Since there is no number held by CNT yet, assign a number to the variable by typing the following: 5 LET CNT = 10 _ _ _ _

NOTE: LET assigns the numeric value of 10 to the variable CNT.

(Listing #4)

```
5 LET CNT = 10
10 REM ***  Program to demonstrate the FOR-NEXT loop
20 CLS
30 INPUT NM$       :REM Get the name from the keyboard.  Start a
40 FOR TIMES = 1 TO CNT  :REM counter called TIMES to count to 5
50 PRINT NM$       :REM Print the string variable on the screen
60 NEXT TIMES          :REM Loop back to FOR
65 REM  Add 1 (INCREMENT by 1) to the counter and
66 REM  print the string variable again.
70 REM  When counter has reached 5, FALL OUT OF THE LOOP
75 REM  and go to next EXECUTABLE line.
80 PRINT "DONE"    :REM Then print DONE on the screen
90 END             :REM Always end your program with END
```

12. Your program should look like Listing #4. LIST and verify that it is correct. _ _ _ _

13. RUN the program. EDIT line 5 and change the value of CNT to a new number. RUN the program. Did the count for TIMES change to match the value of CNT? ____ _ _ _ _

14. Change CNT and RUN the program several times. Does it work? ____ _ _ _ _

15. To quit BASIC and return to DOS, type SYSTEM. Run the program for the instructor. _ _ _ _

16. Turn computer off and store diskettes. _ _ _ _

Average of above evaluations* _ _ _ _

*This average is the student's partial evaluation for competency I5. Write simple computer programs.

WRITE PROGRAM IN BASIC--PT. 2

Equipment:

DOS and GWBASIC diskette (or BASICA)
IBM-compatible computer
File diskette

Procedure 1

		Evaluation 3	2	1	N
1.	Boot the PC. Load BASIC by typing GWBASIC (or BASICA if using an IBM).	—	—	—	—
2.	Until now, lines have been numbered one at a time before each BASIC command. The computer can do this for you! Type AUTO and the computer will number the lines automatically.	—	—	—	—
3.	Enter Listing #1.	—	—	—	—

(Listing #1)

```
10 REM ** Program to demonstrate LOCATE to position the cursor
20 REM ** Add a message line to INPUT.  Use LOCATE DOWN,ACROSS
30 REM ** Example:  LOCATE 10,20 moves the cursor down 10, across 20
40 CLS:LOCATE 10,20:REM Notice multiple statements with colon separator
50 INPUT"What is your name ";NM$ :REM Add the message to INPUT
60 REM ** Now move down 3 lines and print phrases with string variable
70 LOCATE 13,20
80 PRINT"Hello";NM$;"I am glad to know you!"
90 END
```

4.	When the listing is complete, stop the automatic line numbering by pressing the CONTROL and BREAK keys at the same time. RUN the program.	—	—	—	—

NOTE: After clearing the screen, the text was not printed in the upper left corner of the screen. LOCATE told BASIC where to print the text by moving the cursor to row 10 (down) and column 20 (across). The CRT screen is made up of 25 lines or rows and 80 columns or spaces on each line. LOCATE will position the cursor any place on the screen for the next PRINT statement.

5.	Change the LOCATE row and column number to move the cursor to another location. Remember to use EDIT to pull the line down so it can be changed. The change only takes place after ENTER is pressed.	—	—	—	—
6.	Line 80 prints your name jammed between Hello and I with no space between them. (Example: HelloNameI am glad to know you!) Add a space after the "o" of "Hello" and before the "I" to make the text more readable.	—	—	—	—

a. First type EDIT 80 to edit the line.
b. Next, place the cursor where you want the space.
c. Press the Ins (INSERT) key, then the SPACE BAR to insert the space. Repeat for each position where more space is desired.
d. Press ENTER to store the change in memory.

7. RUN the program. Is the printed text more readable now? ____ — — — —

8. Notice that the semi-colons in line 80 held the cursor on the same line. Change the semi-colons to colons and note how the text prints on the screen when the program is run. — — — —

9. Explain the difference in print-out when using the colon and when using the semi-colon. — — — —

Procedure 2

1. Add Listing #2 to Listing #1. — — — —

(Listing #2)

```
90 REM Add these lines to your program in Listing 1.
100 REM Also notice that more than one statement can be
110 REM on one line, separate by a colon(:).
115 FOR SLOW = 1 TO 1000:NEXT SLOW:BEEP
120 LOCATE 23,60:PRINT "First"
125 REM NOW LOCATE TO BOTTOM RIGHT CORNER
130 ' Notice that the ' can be used instead of REM
135 FOR SLOW = 1 TO 1000:NEXT SLOW:BEEP
140 LOCATE 2,5:PRINT "Second" :' Move to top left corner
150 FOR DLA = 1 TO 1000 : NEXT DLA :BEEP
160 LOCATE 23,5:PRINT"Third" :' Now to the bottom left corner
170 FOR DLA = 1 TO 2000 : NEXT DLA :BEEP:' Delay twice as long
180 LOCATE 2,60:PRINT"Fourth" :' Now to the top right corner
190 FOR TOOT = 1 TO 3000:NEXT TOOT
195 LOCATE 25,35 :PRINT"FINISHED !":BEEP
200 LOCATE 5,1:' Put cursor back to top of screen
210 END
```

2. Use the ' as a notation for REM to set up a time delay using the FOR-NEXT loop and BEEP the speaker. Observe the results as the program is run. Try changing the time delay. — — — —

Procedure 3

1. Clear the computer memory. — — — —

2. Type the program in Listing #3. — — — —

(Listing #3)

```
5 REM ** PROGRAM TO CALCULATE TOLERANCE LIMITS OF A RESISTOR **
10 PRINT "ENTER RESISTOR VALUE" ' Print message on screen
20 PRINT "R = "; 'Print message on screen, wait for Input
30 INPUT R  :REM  Wait for value and assign it to variable R
40 PRINT "ENTER TOLERANCE IN DECIMAL FORM, (.05, .1, .2, .02)"
45 REM  Line 40 prints message on screen
50 PRINT "T ="; 'Prints message, waits for input of tolerance
60 INPUT T  'Takes tolerance input assigns to variable T
65 LET TT = T * 100 'Converts decimal value to percent
70 LET R1 = R * (1-T)  'Formula to calculate minimum resistance
75 REM tolerance value (R - the tolerance)
80 LET R2 = R * (1+T)  'Formula to calculate maximum resistance
```

```
85 REM tolerance value (R + the tolerance)
90 PRINT "THE MINIMUM VALUE IS ";R1;" OHMS."
95 PRINT "THE MARKED VALUE IS ";R;" OHMS."
97 PRINT "THE TOLERANCE IS ";TT;" %"
100 PRINT "THE MAXIMUM VALUE IS ";R2;" OHMS."
105 REM  Lines 90 to 100 print out the calculated values
110 GOTO 10  'The GOTO loops back to line 10 to do it again
115 REM To break out of the loop, use CTRL BREAK
120 END  ' Always have an End to your program
```

NOTE: Up to now all your programs have been "straight line" programs. Each line is executed and BASIC goes to the next line in the number sequence until the last numbered line or END is encountered.

Notice line number 110 in the program you just typed. The GOTO tells BASIC to redirect the sequence of execution. In this case BASIC is told to go back to line number 10 and continue execution.

GOTO causes an unconditional branch to another line number. The program is now in a loop because each time BASIC executes line number 110, the program will branch to line number 10 and repeat. At this time the only way out of the loop is to stop the program with CTRL-BREAK.

3. Go through the program line by line. Try to predict what will occur on each line. Observe how variables and math are used. — — — —

4. RUN the program several times. Does it work as predicted? _____ — — — —

5. Add your own instructions to do the following. — — — —

   a. Clear the screen.
   b. Position the cursor near center of the screen for each PRINT and INPUT.
   c. Make an attractive screen display.

6. Run program for the instructor when completed. — — — —

7. Turn off computer and put diskettes away. — — — —

Average of above evaluations* — — — —

*This average is the student's partial evaluation for competency I5: Write simple computer programs.

WRITE PROGRAM IN BASIC--PT. 3

Equipment:

DOS and GWBASIC diskette (or BASICA)
IBM-compatible computer
File diskette

Procedure 1

Evaluation 3 2 1 N

1. Boot the PC. Load BASIC. — — — —

2. Type Listing #1. — — — —

(Listing #1)

```
1 ' Program Name OHMSLAW.BAS
2 '
5 CLS
10 LOCATE 5,15:PRINT "*** PROGRAM TO CALCULATE OHM'S LAW ***"
20 LOCATE 9,27:PRINT "1  Solve for E"
30 LOCATE 10,27:PRINT "2  Solve for I"
40 LOCATE 11,27:PRINT "3  Solve for R"
44 '
45 '  Notice on line 50 a PROMPT or MESSAGE can be use with INPUT
46 '
50 LOCATE 15,18:INPUT "Select by Pressing  1  2  or  3";SEL
60 END
```

NOTE: Listing #1 is an example of a menu with selections for the operator to choose a procedure. The REM or ' is used on some lines to make the listing easier to follow. They may be ignored, but comments or REMs do make a listing easier to understand.

3. Notice the way in which a PROMPT can be added to INPUT within quotes. Placing the semicolon before the variable SEL keeps the cursor and the question mark at the end of the printed message.

4. RUN the program to observe the menu. Note that when a menu selection is made the number value is assigned to SEL but the program ends before anything is done with it. — — — —

Procedure 2

1. This program will be added to until it becomes quite lengthy. To SAVE the program to disk, the drive and a filename must be specified in the form "drive:filename.extender". — — — —

2. Be sure that a formatted data diskette is in drive B and type the following:

   SAVE "B:OHMSLAW.BAS"

   . . . then press return. The extender BAS is a reminder that this is a BASIC file.

3. To get a directory from BASIC, type FILES for a list of files on the default disk. Try it. — — — —

4. Type FILES "B:" to see the files on drive B. Is your program there? ____ _ _ _ _
If the program was saved safely, exit BASIC by typing SYSTEM.

NOTE: This job sheet will introduce two ways to load a disk file into BASIC.

5. From DOS type GWBASIC B:OHMSLAW.BAS then press return. BASIC should start, LOAD the program file, then RUN it in one operation. Try it. Does it "come up" running? ____ _ _ _ _

6. Type SYSTEM to get back to DOS and try the second method. From DOS type GWBASIC to start BASIC. When the OK prompt comes up, type LOAD "B:OHMSLAW.BAS". When the file is loaded, the OK prompt will return. Now type LIST or RUN to continue. _ _ _ _

Procedure 3

1. LIST the program on the screen. Notice that the user must make a selection of 1, 2 or 3 and then assign the number to the variable SEL (for select). The program ends before anything is done with the number. _ _ _ _

2. The IF-THEN pair is used for decision-making. Note the format of the following lines. They are true or false decisions. _ _ _ _

(Listing #2)

```
60 IF SEL = 1 THEN GOTO 200
70 IF SEL = 2 THEN GOTO 400
80 IF SEL = 3 THEN GOTO 600
90 END
```

NOTE: Line 60 says that IF the variable SEL is equal to 1, THEN GOTO line 200. IF the variable SEL IS NOT EQUAL to 1, ignore the rest of line 60 and drop through to execute line 70.

On line 70, IF the variable SEL is equal to 2, THEN GOTO line 400. IF the variable SEL IS NOT EQUAL to 2, ignore the rest of line 70 and drop through the loop to execute line 80.

On line 80, IF the variable SEL is equal to 3, THEN GOTO line 600. IF the variable SEL IS NOT EQUAL to 3, ignore the rest of line 80 and drop through to execute line 90.

3. RUN the program and observe the operation. Try selections 1, 2 and 3. What happens?

Now try other numbers. What happens?

4. Add the following lines to the program. _ _ _ _

(Listing #3)

```
85 '
90 BEEP:GOTO 50 ' If 1, 2 or 3 is NOT selected, BEEP and repeat from
50
95 '
100 END 'Use END or program could run through next lines and CRASH!
```

5. If any number other than 1, 2 or 3 is selected, the computer will beep and loop back to line 50. This is a conditional loop that can be broken out of under any one of three conditions! _ _ _ _

6. The menu selections are not "bulletproof". Try selecting an alpha character such as A or B. What happens? — — — —

7. Save the modified program before continuing. A good "rule of thumb" is to save . . . and save often. — — — —

8. Now add Listing #4 to your program. — — — —

(Listing #4)

```
199 '
200 CLS '  Start of routine to solve for E
210 LOCATE 5,25:PRINT"SOLVING FOR VOLTAGE"
220 LOCATE 7,25:INPUT "Enter the value of I";I
230 LOCATE 9,25:INPUT "Enter the value of R";R
240 LET E = I * R  ' Formula for voltage, E, I, R are the variables
250 LOCATE 11,25:PRINT"The value of E is ";E;" Volts"
260 STOP
```

9. Selection of 1 will cause a conditional branch to line number 200. The screen will be cleared; values for variable I for current and R for resistance will be entered.

10. Line number 240 multiplies the value of I by the value of R and assigns the resulting value to the variable E. Now print the contents of the variable E.

11. RUN the program. Note that when the voltage value prints, the program ends. You must RUN the program again to do another problem. — — — —

12. Add Listing #5 to the program. — — — —

(Listing #5)

```
399 '
400 CLS '  Start of routine to solve for I
410 PRINT"SOLVING FOR CURRENT"
420 INPUT "Enter the value of E";E
430 INPUT "Enter the value of R";R
440 LET I = E / R
450 PRINT"The value of I is ";I;" Amperes"
460 STOP
599 '
600 REM  Start of routine to solve for R
610 PRINT"SOLVING FOR RESISTANCE"
620 INPUT "Enter the value of I";I
630 INPUT "Enter the value of E";E
640 LET R = E / I
650 PRINT"The value of R is ";R;" Ohms"
660 STOP
```

13. RUN the program several times. Now make your own changes to make the screens look attractive. Don't forget to save the program. — — — —

Procedure 5

1. The next listing adds a choice to quit the program from the main menu and a choice to repeat a problem or return to the main menu. Add Listing #6 to the program. — — — —

(Listing #6)

```
44 ' Add the choice to QUIT to the Menu
45 LOCATE 12,27:PRINT "4  QUIT the PROGRAM"
46 '
50 LOCATE 15,18:INPUT "Select by Pressing  1  2  3 or 4";SEL
85 IF SEL = 4 THEN GOTO 95
95 CLS :'  Tidy up the screen then end the program!
100 END 'Always use END or program could CRASH!
```

NOTE: A fourth choice has been added to the main menu. If 4 is selected to quit the program, the branch is to line number 95 where the screen is cleared. Line 100 is a logical program end.

2. After running and trying the program, save it to disk and add Listing #7. — — — —

(Listing #7)

```
255 '
260 LOCATE 15,15:INPUT"Do another voltage problem?  Y or N ";YN$
265 '
270 IF YN$ = "Y" OR YN$ = "YES" THEN GOTO 200
280 IF YN$ = "y" OR YN$ = "yes" THEN GOTO 200
285 ' Test for upper case Y, lower case y, and yes
290 GOTO 5 :' If NOT yes it must be NO, so go back to main menu
```

3. After the voltage problem is solved, a prompt is displayed to be answered Y or N. Use a string variable represented by YN$, then the True-False logic of IF THEN. — — — —

4. There could be four possible ways to enter Yes--uppercase Y, YES, lowercase y or yes. Observe closely how lines 270 and 280 test for each of the four possibilities. Notice that line 290 will branch to line 5 for N, n, no or NO. — — — —

5. Save your complete listing, then run the program. — — — —

6. Add your own code to the listing so that the current and resistance routines offer the choice between another problem or returning to the main menu. Use Listing #7 as a guide.

7. Test it, save it, then show the program to the instructor. — — — —

8. Turn off computer power and put diskettes away. — — — —

Average of above evaluations* — — — —

*This average is the student's partial evaluation for competency I5: Write simple computer programs.

WRITE PROGRAM IN BASIC--PT. 4

Equipment:

DOS and GWBASIC diskette (or BASICA)
IBM-compatible computer
File diskette

Procedure 1

Evaluation 3 2 1 N

1. Boot the PC. — — — —

2. Load BASIC. — — — —

3. Now a "for fun" option. Enter Listing #1. — — — —

(Listing #1)

```
10 CLS
20 LET NM$ ="YOUR NAME HERE"
30 LET TITLE$ = "RESISTOR CALCULATOR"
40 FOR MOVIT = 1 TO 25
50 LOCATE 2,MOVIT
60 PRINT TITLE$
70 FOR DLA = 1 TO 200:NEXT DLA
80 NEXT MOVIT
85 '
90 LOCATE 3,34:PRINT "by"
```

NOTE: Between lines 40 and 80 a counter is created with a FOR NEXT loop and the variable MOVIT. The LOCATE is down two rows, but the column number is determined by the current number in the counter MOVIT. Each time the counter increments, line 60 prints the string in a different horizontal position.

3. Line 70 is a "do nothing" loop to kill time before printing the string again. Change the speed by changing the count for DLA. RUN the program and observe the result. — — — —

4. Did you get a string of trailing R's while TITLE$ moved across the screen? _____ Can this be corrected? _____ Try it! — — — —

   HINT: print a NULL (space) to the left of the R.

5. When the display has been cleaned up, enter Listing #2. — — — —

(Listing #2)

```
100 FOR MOVIT = 65 TO 30 STEP -1
110 LOCATE 4,MOVIT
120 PRINT NM$
130 FOR DLA = 1 TO 150:NEXT DLA
140 NEXT MOVIT
```

6. This time the counter has been told to decrement or count backwards with the STEP -1. Use your name for NM$. Change the values if needed to center the strings on the screen. — — — —

Procedure 2

1. Enter Listing #3 to add a menu to the screen. — — — —

(Listing #3)

```
150 LOCATE 8,25:PRINT "Series Resistors"
160 LOCATE 10,24:PRINT "Parallel Resistors"
170 LOCATE 13,24:INPUT"Choose  S  or  P";G$
180 IF G$ = "s" OR G$ = "S" THEN GOTO 220
190 IF G$ = "p" OR G$ = "P" THEN GOTO 330 ELSE GOSUB 460
```

NOTE: This example uses a string variable for menu selection instead of a numeric variable. Lower case and upper case are tested. Line 180 tests for upper and lower case S. If TRUE, a branch to line 220 occurs. Lines 220, 330 and 460 are undefined at this time. If FALSE, the program executes line 190. Line 190 tests for upper and lower case P. If TRUE, a branch to line 330 occurs. If FALSE, ELSE causes a branch to line 460.

The branch is executed by GOSUB. GOSUB is different from GOTO--GOSUB must always be paired with a RETURN instruction.

2. RUN the program to observe execution. — — — —

3. Now add Listing #4 to observe a subroutine called by GOSUB. A GOSUB RETURN subroutine is used where a procedure in a program can be repeated often in exactly the same way. Instead of writing the same instructions, many times the subroutine can be written one time, then called when needed. Both memory space and programmer's time are saved. — — — —

(Listing #4)

```
460 BEEP :'  EXAMPLE OF A SUBROUTINE
470 LOCATE 24,10:PRINT"You selected the wrong letter!";
480 FOR DLA = 1 TO 500:NEXT DLA
490 LOCATE 24,10:PRINT"                                    "
500 BEEP
510 RETURN :'  End of subroutine, return to main program
```

NOTE: When the subroutine is executed, a message is printed on the screen and the beep alerts the operator. Line 480 keeps the message on the screen for a short time, then line 490 erases the message with a string of spaces or NULLS. RETURN branches back to the end of line 190. Since there are no further instructions on line 190, line 200 is executed.

4. Add Listing #5 to the program. — — — —

(Listing #5)

```
200 CLS:LOCATE 5,25:PRINT" RESISTOR CALCULATOR"
210 GOTO 150
```

5. What should lines 200 and 210 do? — — — —

6. Try it! Did the program perform as expected? ____ — — — —

7. Examine Listing #6. With pencil and paper go through the routine step by step, jotting down what is happening on each line. Assume that four resistors will be calculated. — — — —

8. Explain to the instructor what you determined in Step 7. — — — —

9. Enter Listing #6. — — — —

(Listing #6)

```
220 'Program to calculate Series Resistors
230 CLS:LOCATE 2,15:PRINT"SERIES RESISTORS"
240 LOCATE 4,10:INPUT"How many resistors are there";N
250 FOR CNT = 1 TO N
260 LET DN = 10 + CNT
270 LOCATE DN,10:PRINT"Enter the value of resistor number ";CNT;" ";
280 INPUT;R
290 LET RT = RT + R
300 NEXT CNT
310 LOCATE DN + 2,10:PRINT"The total value Rt = ";RT;" Ohm"
320 STOP
```

10. RUN the program. Did it perform as expected? _____ — — — —

11. Examine Listing #7. With pencil and paper go through the routine step by step, jotting down what is happening on each line. Assume that four resistors will be calculated. — — — —

(Listing #7)

```
330 'Program to calculate Parallel Resistors
340 CLS:LOCATE 2,15:PRINT"PARALLEL RESISTORS"
350 LOCATE 4,10:INPUT"What is the value of resistor #1 ";R
360 LET RT = R
370 LOCATE 6,10:INPUT"How many resistors are there";N
380 FOR CNT = 2 TO N
390 LET DN = 10 + CNT
400 LOCATE DN,10:PRINT"Enter value of resistor number ";CNT;" ";
410 INPUT;R
420 LET RT = (RT * R) / (RT + R)
430 NEXT CNT
440 LOCATE DN + 2,10:PRINT"The total value Rt is ";RT;" Ohms"
450 STOP
```

12. Remembering basic DC theory, which method of solving parallel resistors is used in this routine? _____ — — — —

13. Explain to the instructor what you determined in Step 12. — — — —

14. Enter Listing #7 and RUN the program. Did it perform as expected? _____ — — — —

Procedure 3

NOTE: The command LIST has been used so far to list the program to the screen. The program can be listed to a printer by using the command LLIST.

1. Connect a printer to the computer. When ready and while still in BASIC, type LLIST for a hard copy of the program. — — — —

2. Run the program for the instructor. — — — —

3. Turn off computer and put diskettes away. ___ ___ ___ ___

Average of above evaluations* ___ ___ ___ ___

*This average is the student's partial evaluation for competency I5. Write simple computer programs.

DEBUG SOFTWARE

Equipment:

BASIC language
IBM PC or compatible
File diskettes
BASIC program written during JS 2.1-2.4

Procedure:	Evaluation 3	2	1	N
1. Add a routine to the end of the series resistors routine to do another series resistor problem or to return to the main menu.	—	—	—	—
2. Add a routine to the end of the parallel resistors routine to do another parallel resistor problem or to return to the main menu.	—	—	—	—
3. Add a selection to the Main Menu of: Q to Quit the program	—	—	—	—
4. Change any LOCATE positions in the program to make an attractive, balanced screen.	—	—	—	—
5. If a color display is available, add color to the program. Choose color combinations that are attractive and easy to read.	—	—	—	—
6. Add some sound, if desired.	—	—	—	—
7. Show program to the instructor when completed.	—	—	—	—
8. Turn off power and put diskettes away.	—	—	—	—
Average of above evaluations*	—	—	—	—

*This average is the student's final evaluation for competency I6: Debug software.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT III: COMPUTER PROGRAMMING

UNIT TEST

Select the option which answers the question or completes the statement.

___ 1. What is always used with a RETURN instruction in BASIC?

a. GOTO
b. GOSUB
c. PRINT
d. SAVE
e. INPUT

___ 2. What does a $ indicate in BASIC?

a Variable
b. String variable
c. End of an ASCII string
d. Dollars and cents
e. A $ is never used.

___ 3. What are binary numbers that the microprocessor understands called?

a. Object code
b. Source code
c. Compiler
d. Interpreter
e. Hexadecimal

___ 4. What reads the program one line at a time and converts it to binary code?

a. Object code
b. Source code
c. Compiler
d. DOS
e. Interpreter

___ 5. What converts the program to a stand-alone binary program that will run without the language?

a. Object code
b. Source code
c. Compiler
d. Interpreter
e. DOS

___ 6. What are instructions written by the human operator called?

a. Object code
b. Source code
c. Compiler
d. Interpreter
e. None of the above

___ 7. Which of the languages listed is not a high-level language?

a. BASIC
b. Assembler
c. Pascal
d. Fortran
e. COBOL

___ 8. Which is not a characteristic of a low-level language?

a. Program runs faster
b. Program binary code is small
c. Takes less space in memory
d. Uses English statements
e. None of the above

___ 9. Which language is most often found in the interpreter form but can also be a compiled language?

a. BASIC
b. C
c. Pascal
d. Fortran
e. COBOL

___ 10. What is true of BASICA or GWBASIC?

a. BASICA is the advanced BASIC for only IBM PC computers.
b. GWBASIC is a lower-level language for IBM-compatibles.
c. Part of GWBASIC's language is in ROM memory.
d. All of BASICA can be loaded from disk into memory.
e. BASICA and GWBASICA can be used interchangeably.

___ 11. Which symbol is used for mathematical calculations in BASIC?

a. +
b. -
c. *
d. /
e. All of the above

___ 12. Which of the following symbols is not used in BASIC programming?

a. +
b. -
c. *
d. #
e. ^

___ 13. Which describes a purpose or aspect of a GOSUB ... RETURN command?

a. Branches to and returns from a subroutine
b. Performs a series of instructions in a loop
c. Loops a given number of times
d. Variable to be used as a counter
e. x = initial value of the counter

Match the following terms and definitions.

____ 14. Reserved place in memory that holds a numeric value

____ 15. Advanced BASIC only on IBM-PC computers

____ 16. Text of alphabetic characters or numbers without a numeric value

____ 17. Generic, advanced BASIC for IBM-compatibles

____ 18. Math operations using positive or negative numbers with fractional parts or only decimal fractions

____ 19. Reserved place in memory that holds text characters; enclosed by quotes

a. BASICA

b. Constant

c. Floating point division

d. GW-BASIC

e. Integer division

ab. String

ac. String variable

ad. Variable

ANSWERS TO UNIT TEST

1.	b	(I5)
2.	b	(I5)
3.	a	(I2)
4.	e	(I2)
5.	c	(I5)
6.	b	(I5)
7.	b	(I2)
8.	d	(I2)
9.	a	(I2)
10.	a	(I5)
11.	e	(I5)
12.	d	
13.	a	(I5)
14.	ad	(I5)
15.	a	(I5)
16.	ab	(I5)
17.	d	(I5)
18.	c	(I5)
19.	ac	(I5)

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT IV: MICROPROCESSOR INSTRUCTIONS

UNIT OBJECTIVE

Upon completion of this unit, the student should be able to write, analyze and debug machine language programs. This knowledge will be evidenced by completion of the job sheets to specified score

SPECIFIC OBJECTIVES

After completion of the lessons in this unit, the student should be able to:

1.1 Define terms and definitions for the unit. (AS 3.1)

1.2 Write assembly language programs using 8088 instructions. (JS 1.1)

1.3 Write a program in assembly language using DEBUG. (JS 1.1)

2.1 Demonstrate flags in programming. (JS 2.1)

2.2 Write jump and loop programs. (JS 2.1)

3.1 Use common addressing modes. (JS 3.1)

4.1 Write and assemble stand-alone assembly language .COM files. (JS 4.1)

5.1 Use shift, rotate and test instructions. (JS 5.1)

5.2 Use Boolean instructions to mask, set and toggle individual bits. (JS 5.1)

5.3 Use IN and OUT instructions to I/O port. (JS 5.1)

5.4 Write, debug and run an original assembly language program. (competencies I4-I6, JS 5.2)

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT IV: MICROPROCESSOR INSTRUCTIONS

CONTENTS OF THIS UNIT

A. Objective sheet

B. References

C. Notes to the instructor

D. Lesson plans

1. Lesson 1: BEGINNING 8088 PROGRAMMING

   a. Information outline

   b. Handouts

      HO 1.1: ASCII code
      HO 1.2: Memory map

   c. Job sheet

      JS 1.1: Write program in assembly language

2. Lesson 2: DECISION-MAKING PROGRAMMING

   a. Information outline

   b. Handout

      HO 2.1: Conditional jump instructions

   c. Job sheet

      JS 2.1: Demonstrate flags and loops

3. Lesson 3: ADDRESS MODES

   a. Information outline

   b. Assignment sheet

      AS 3.1: Terms and definitions

   c. Answers to assignment sheet

   d. Job sheet

      JS 3.1: Use addressing modes

4. Lesson 4: MORE MATH OPERATIONS

   a. Information outline

b. Job sheet

JS 4.1: Compute math operations

5. Lesson 5: ADVANCED PROGRAMMING

a. Information outline

b. Job sheets

JS 5.1: Demonstrate advanced programming
JS 5.2: Analyze simple programs

REFERENCES

A. Books

1. Brey, Barry B. 8086/8088 Microprocessor: Architecture, Programming, and Interfacing. Columbus, OH: Merrill, 1987.

2. Brey, Barry B. The Z80 Microprocessor: Hardware, Software Programming and Interfacing. Englewood Cliffs, NJ: Prentice-Hall, 1988.

3. Coffron, James W. Microprocessor Programming, Troubleshooting, and Interfacing: The Z80, 8080, and 8085. Englewood Cliffs, NJ: Prentice Hall, 1988.

4. Dailey, Denton J. Small Computer Theory and Applications. New York: McGraw-Hill, 1988.

5. Gaonker, Ramesh. The Z80 Microprocessor: Architecture, Interfacing, Programming, and Design. Columbus, OH: Merrill, 1988.

6. Gillmore, Charles M. Microprocessors: Principles and Applications. New York: McGraw-Hill, 1989.

7. Hall, Douglas V. Microprocessors and Interfacing: Programming and Hardware. New York: McGraw-Hill, 1986.

8. Harman, Thomas L. The Motorola MC68020 and 68030 Microprocessors: Assembly Language, Interfacing, and Design. Englewood Cliffs, NJ: Prentice-Hall, 1989.

9. Hughes, Fredrick W. Microprocessor Technology: Theory and Experimentation. Englewood Cliffs, NJ: Prentice-Hall, 1989.

10. Kersey, Roger M. Personal Computer Operation and Troubleshooting. Englewood Cliffs, NJ: Prentice Hall, 1989.

11. Lafore, Robert. Assembly Language Primer for the IBM PC & XT. New York: The Waite Group, 1984.

12. Miller, Michael A. The 68000 Microprocessor: Architecture, Programming, and Applications. Columbus, OH: Merrill, 1988.

13. Norton, Peter. Inside the IBM PC. New York: Brady, 1986.

14. O'Connor, Patrick Joseph. Digital and Microprocessor Technology. 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1989.

15. Pasahow, Edward J. Microprocessor Technology and Microcomputers. New York: McGraw-Hill, 1988.

16. Tocci, Ronald J., and Lester P. Laskowski. Microprocessors and Microcomputers: Hardware and Software. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1987.

B. Curriculum guides

1. Brown, A. O. Advanced Microcomputer Service Technician. Stillwater, OK: Mid-America Vocational Curriculum Consortium, 1985.

NOTES TO THE INSTRUCTOR

A. Suggested approaches to teaching this unit

1. Demonstrate DEBUG commands.
2. Explain each microprocessor instruction.
3. Demonstrate use of each instruction with DEBUG.

B. Suggested student activities

1. Enter and assemble job sheet examples using HO 1.1 where appropriate.
2. Change and add to the examples.
3. Students should write their own programs with DEBUG.

C. Suggested supplemental material (postsecondary)

1. Brey, Barry B. 8086/8088 Microprocessor: Architecture, Programming, and Interfacing. Columbus, OH: Merrill, 1987.

2. Hall, Douglas V. Microprocessors and Interfacing: Programming and Hardware. New York: McGraw-Hill, 1986.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT IV: MICROPROCESSOR INSTRUCTIONS

LESSON 1: BEGINNING 8088 PROGRAMMING

NOTE: Upon completion of this lesson, the student should be able to program the 8088 to add, subtract, multiply, divide and print ASCII text.

I. Terms and definitions

A. ASCII (American Standard Code for Information Interchange)--use of seven bits to code up to 128 uppercase and lowercase alphabet characters, numbers, punctuation marks and control characters. Using eight bits can code an additional 128 special characters.

B. Hexadecimal--base 16 number system using numbers 0-9 and letters A-F; abbreviated „

C. Interrupt--function call asking DOS to use a built-in assembly language routine to do something for the programmer; saves the programmer time since the programmer does not have to write the routine

D. Radix point--generic term for decimal point

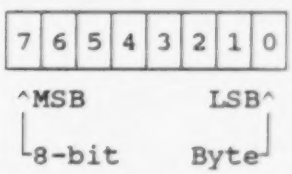
II. Inside the 8086/88

A. Bits and bytes

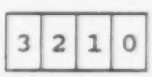
1. For 8086/88 systems, binary digits are arranged in groups of 8-bits called a byte.

2. The least significant bit (LSB) is usually shown on the right and labeled bit 0.

3. The most significant bit (MSB) is shown on the left and is labeled bit 7.



4. Four bits (half a byte) are called a nibble.



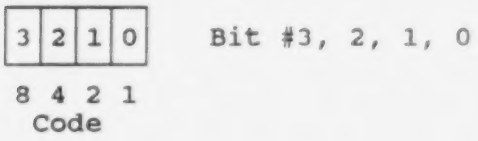
4-bit nibble

5. The four bits of the nibble and the weight of each bit position is known as the 8421 code.

6. The total number of possible bit patterns is 2^4 or 16 (decimal).

7. The 16 patterns go from 0000b to 1111b. (b indicates binary.)

8. The 16 possible bit patterns range from 0 to 15d (decimal).



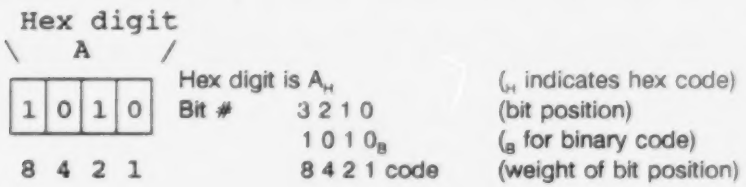
9. Binary numbers are difficult to use or even speak.

10. To make it easier to use, binary code is converted to the hexadecimal number system.

11. Decimal numbers range from 0 to 9 as single digits, then carry 1 and go back to 0 for decimal 10.

12. To have 16 single digit numbers (including 0), the letters A through F are used as numbers.

13. The hexadecimal digits then are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

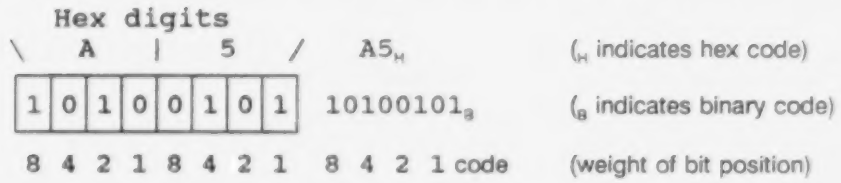


14. A full byte is handled by breaking the byte into two nibbles.

15. Each nibble is a hex digit.

16. An 8-bit binary number then is expressed as two hex digits.

17. Hex coding is only for operator convenience--the microprocessor still uses binary.



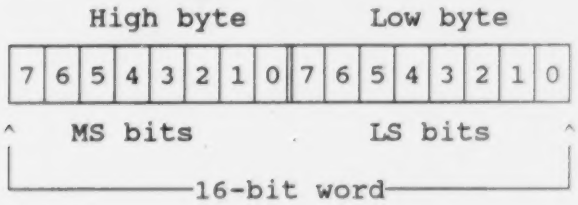
B. Words

1. Two bytes represent a 16-bit pattern called a word.

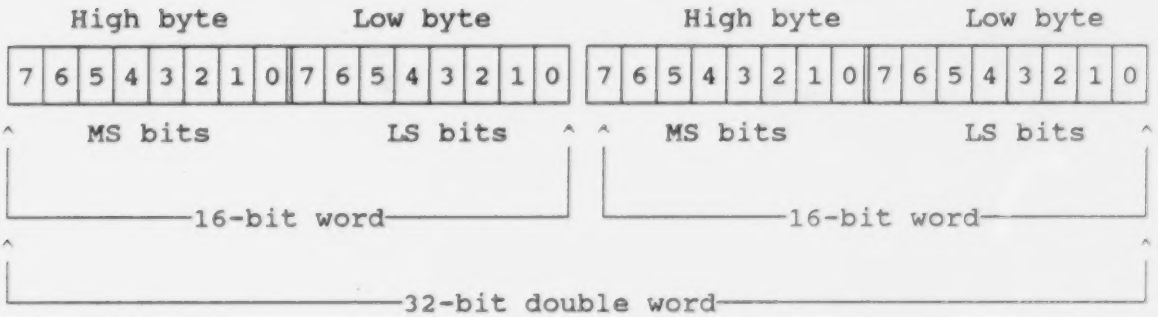
2. The byte containing the LSB is the low-order byte.

3. The byte containing the MSB is the high-order byte.

4. Four hex digits can represent the 16 bits (0000_H to $FFFF_H$).



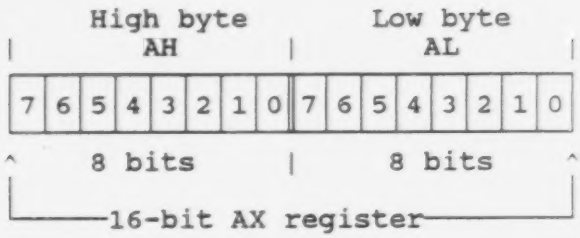
5. Some of the new generation of microprocessors handle 32-bits as a double word.


High byte
Low byte
7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
MS bits
LS bits
16-bit word
High byte
Low byte
7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
MS bits
LS bits
16-bit word
32-bit double word

C. Registers

1. Every microprocessor has groups of flip-flops constructed inside of the chip.
2. The groups of flip-flops are called registers.
3. Some registers are one byte long; some are two bytes long.
4. A register is not a memory location, but it does temporarily store information inside the microprocessor.
5. A register is similar to a variable in BASIC.
6. A register is a temporary storage place inside the microprocessor.
7. The 8086/88 microprocessor has 16-bit registers.
8. The contents of a register, like memory, can be expressed as hex digits.
9. Unlike memory, a register does not have an address. Instead, the register has a name that identifies it and the general use of that register.
10. Four of the registers are general purpose, while some are rather specialized.
11. The four-general purpose registers are called AX, BX, CX and DX.

12. The AX register is shown below.



13. Some of the registers can be used as two eight-bit registers, while others are strictly 16-bit registers.

14. Notice that the register above is called the AX register when used as a 16-bit register. It can also be used as two 8-bit registers.

15. The low-order byte is then called the AL register, and the high-order byte is called the AH register.

D. Memory

1. Storage of information in the microcomputer is accomplished by essentially three types of memory: random-access memory (RAM), read-only memory (ROM) and mass storage.

2. Mass storage is external to the microcomputer, usually cassette tape, floppy disk or hard disk.

3. Think of RAM and ROM as a series of letter boxes. The contents of the box is the information (data or instructions) to be stored or "remembered".

4. The location of the letter box is the address. There are no duplicate addresses.

5. Information is written (stored) in memory. If the location already has information, it is replaced with the new information.

6. To read or retrieve information, supply the location address and read the stored information.

7. Random-access memory stores information as long as power is applied to the microcomputer.

8. RAM is read/write (R/W) memory.

9. Information can be stored both at an address and read back.

10. Read-only memory already has information stored in each address. The information can only be read or retrieved.

11. Groups of flip-flops make up the electronic construction of both RAM and ROM.

12. When a flip-flop is set, it stores a binary 1. When reset, the flip-flop stores a binary 0.

13. Each binary digit is a binary bit.

14. The flip-flops are arranged in groups of eight bits or bytes.

E. Address segmentation

1. Four binary digits can be written as one hexadecimal digit.
2. A byte then is represented by two hex digits.
3. A word is represented by four hex digits.
4. 2^{16} binary digits imposes a limit of 65,535 memory locations (usually referred to as 64K).
5. The 8088 can actually access over 1M (megabyte) of memory.
6. This is done by adding two numbers together with one of the numbers offset by four binary digits. In effect, the sum is a 20-digit binary number.
7. Two registers in the 8086/88 hold the NEXT memory address that the microprocessor will go to.
8. The CS register holds the number of a 64K segment. Each segment begins at a multiple of 16 bytes.
9. The IP (instruction pointer) register holds the address or OFFSET into the segment, or the byte in the segment.
10. The absolute address is written $0EAF{:}0100_H$.
    a. The number on the left of the colon is the segment address.
    b. The number on the right of the colon is the offset into the segment.
11. The actual address is given as SEGMENT:OFFSET.

    Example: $0EAF{:}0100_H$
    ↑ ↑
    Segment Offset
12. The segment address is stored in register CS, the code segment register.
13. The offset address is stored in the IP register.
14. For now, when looking at an address we will use only the offset.
15. The actual segment in the PC is determined by the amount of memory in the PC, DOS size (in bytes) and any resident utilities.

F. Memory map **(HO 1.2)**

1. The 8086/88 microprocessor expects to find RAM memory at the bottom (low addresses) and ROM memory at the top (high addresses) of memory.
2. The exact manner in which this memory is allocated depends upon the manufacturer of the specific computer.
3. The IBM-PC has memory organized in a specific pattern. MS-DOS and PC-DOS are written to use this pattern.
4. Any manufacturers that wants their computer to be compatible with the IBM-PC must use the same pattern.

5. Memory map refers to the pattern or organization of memory.

6. The total amount of RAM installed in a particular machine may vary from 64K to 640K.

7. Handout HO 1.2 shows the memory map of the IBM-PC and compatibles.

8. Addresses are in hexadecimal as noted by the $_H$ after each address number. Addresses are absolute values; they are not given in the segment:offset form.

9. Starting at the bottom of memory (address 0000_H) are 256 interrupt vectors. A vector is the number held in an address that represents another address.

10. The microprocessor will "look" at an address here, get the address held inside the memory location, then look at the new address.

11. Each vector is four bytes (two words) long.

    a. Vector 0 is stored at memory locations 0000_H-0003_H.

    b. Vector 1 is at locations 0004_H-0007_H.

    c. Vector 255 is found at locations $33FC_H$-$3FF_H$.

    d. Each vector is two words long. (Ex.: segment:offset)

12. Next are two "scratchpad" areas--a 256-byte BIOS scratchpad starting at 0400_H and a 256-byte DOS scratchpad starting at 0500_H.

13. These scratchpad areas are reserved by DOS, and the interrupt vector area is reserved by the microprocessor for special use. Change things in here at your own risk!

14. When the PC is booted up, DOS loads into memory starting at 0600_H.

15. The number of bytes of RAM that is needed by DOS depends upon the DOS version and whether any RAM-resident utilities such as "Sidekick," a RAM disk program or background print spooler are present.

16. Version 2.1 of DOS by itself loads to about 6000_H.

17. Immediately above DOS is the transient program area. This space is for .COM or .EXE files (machine language) such as FORMAT, CHKDSK or DISKCOPY to use.

18. When the transient program ends, it gives up the area for the next program to use.

19. From the top of any programs loaded into the transient program area to the top of installed RAM is the user area. This is the area used when writing assembly language programs in DEBUG.

20. The ROM area is reserved for installed ROM memory that holds the monitor or executive programs to help the operator get the PC up and running.

21. The BIOS (Basic Input Output System) is also in ROM. Content of these memory locations can be read but not written to.

22. When the power is turned on, these programs are ready to execute.

III. The DOS debugger (DEBUG)

A. DEBUG is an assembly language program included on the DOS diskette.

B. DEBUG has a mini-assembler that will convert an assembly language program into a binary .COM file that the 8086/88 understands.

C. Other, more powerful (and more difficult) assemblers are also available.

1. Microsoft's MASM--expensive, takes much more memory, will do more, but is harder to learn

2. Borland's Turbo Assembler (TASM)--costs less and is a little more difficult to learn

3. At least 512K of memory needed to run debugger programs for both MASM and TASM

D. It is impossible to see inside the silicon chip called the 8088 or 8086 microprocessor. Only the results of the microprocessor's operations can be seen.

E. DEBUG will display the contents of the 8086/88 registers on the screen for viewing.

F. The older, traditional single-board microprocessor trainer would allow viewing of the contents of one register or one memory location at a time.

G. DEBUG will show the contents of all the registers at one time. The contents of a large number of memory locations can be viewed with DEBUG.

H. The Job Sheet in Unit I was an introduction to DEBUG and some of the powerful commands available to the user.

I. Follow these steps to start DEBUG.

1. Boot the PC with a system DOS diskette.

2. View the directory. The debug program is listed as DEBUG.COM.

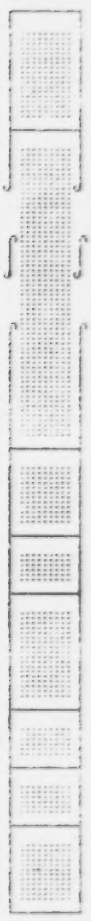
3. Type DEBUG from the DOS prompt to start the program.

	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 A	1011 B	1100 C	1101 D	1110 E	1111 F
000 0	↑@ 0 NUL	↑A 1 SOH	↑B 2 STX	↑C 3 ETX	↑D 4 EOT	↑E 5 ENG	↑F 6 ACK	↑G 7 BEL	↑H 8 BS	↑I 9 HT	↑J 10 LF	↑K 11 VT	↑L 12 FF	↑M 13 CR	↑N 14 SO	↑O 15 SI
001 1	↑P 16 DLE	↑Q 17 DC1	↑R 18 DC2	↑S 19 DC3	↑T 20 DC4	↑U 21 NAK	↑V 22 SYN	↑W 23 ETB	↑X 24 CAN	↑Y 25 EM	↑Z 26 SUB	↑[27 ESC	↑\ 28 FS	↑] 29 GS	↑↑ 30 RS	↑← 31 VS
010 2	32 SP	33 !	34 "	35 #/£	36 $	37 %	38 &	39 '	40 (	41)	42 *	43 +	44 ,	45 –	46 .	47 /
011 3	48 Ø	49 1	50 2	51 3	52 4	53 5	54 6	55 7	56 8	57 9	58 :	59 ;	60 <	61 =	62 >	63 ?
100 4	64 @	65 A	66 B	67 C	68 D	69 E	70 F	71 G	72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O
101 5	80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W	88 X	89 Y	90 Z	91 [	92 \	93]	94 ↑	95 ←
110 6	96	97 a	98 b	99 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
111 7	112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w	120 x	121 y	122 z	123 {	124 |	125 }	126 ~	127 DEL

ASCII CODE

HO 1.1

MEMORY MAP



Top of ROM — FFFFH

ROM Starts

Top of Installed RAM

User Area Starts

Transient Program Area

Resident Utilities — 6000H (Approximately)

DOS — 0600H

DOS Data — 0500H

BIOS Data — 0400H

03FFH

Interrupt Vectors — 0000H

WRITE PROGRAM IN ASSEMBLY LANGUAGE

NOTE: This job sheet uses DEBUG commands E, U, A, T, G, D, N and W, as well as several immediate register mode instructions. The exercise's evaluation is found at the end of the job sheet.

Equipment:

DOS master diskette with DEBUG.COM
IBM-compatible computer
Diskettes
HO 1.1: ASCII code

Procedure 1: Add registers

1. To add the contents of BX to the contents of AX, use ADD AX,BX. This is a two-byte instruction, $01D8_H$, that will have to reside in the computer memory.

2. To place the two-byte instruction in the computer memory, use the DEBUG command E (Enter). Start DEBUG, then give the command U (Unassemble) to verify that the memory locations starting at 0100_H contain "garbage". At the - prompt, give the command E 100 (tells DEBUG to enter something at address 0100_H). DEBUG will respond with the CS:IP address followed by the contents of that address and a period, display the - prompt and wait for the operator to enter the new numbers for that address.

   Example: 0EAF:0100 39.-

   Type 01 and Enter. Type E 101 (for offset address 0101_H), then at the prompt type D8.

3. Give the Register display command. Recall that DEBUG starts by setting the IP to 0100_H, the address where the first instruction byte was stored. At the bottom row of the register display you should see:
   `xxxx:0100 01D8 ADD AX,BX .`

   The op code 01D8 has been interpreted by DEBUG to mean "add the contents of BX to the contents of AX." Notice the "slightly backward" logic of this command. The last register given is added to the first register given. The Unassemble command can be used to view this one-byte program.

4. Store 1111_H in AX with the Register command. Store 2222_H in BX with the Register command. Display the registers to verify that AX and BX hold your numbers, that IP is at 0100_H and that addresses 0100_H and 0101_H hold your op code.

5. Without stop or halt instructions following the op code, the program will start executing without restraints. The DEBUG command T (Trace) executes one command at a time and then displays the registers. Type T. Did it work? ____

6. What number does AX hold? ____ What number does BX hold? ____ Note that the old contents of AX are destroyed and replaced with the sum of BX and AX. To add the present contents of BX to the present contents of AX, what would be the new sum? _____ What number does IP hold? ____

7. If the T command were given at this time, the 8088 would start executing at address 0102_H. Since that is not part of our program, the PC may act peculiarly. Make sure that the IP is set where execution should start. Use the R command to set IP back to 0100_H. Execute the program again using the T command. AX now holds _____. Try adding several different numbers by placing new numbers in AX and BX. _____

8. Store 1111_H in AX with the R command. Store 2222_H in BX with the R command. Display the registers to verify that AX and BX hold your numbers, that IP is at 0100_H and that address 0100_H and 0101_H holds your op code. Trace one time so AX holds 3333_H and BX holds 2222_H. Verify AX _____ and BX _____ .

Procedure 2: Subtract registers

1. The op code to subtract BX from AX and leave the result in AX (SUB AX,BX) is $29D8_H$. This form is similar to AX = AX - BX in BASIC. When the ADD instruction was entered, you used the E command to place one byte in 0100_H and the E command again to place the second byte in 0101_H. Both bytes can be entered on one E command in the following manner. Type -E 100.

2. When DEBUG responds, the old 01D8 op code will still be held in 0100_H and 0101_H, so the response will be 01.-. Type 29 then SPACE and DEBUG responds with the contents of the next address D8.-, then enter D8. Try this a few times.

3. Issue the R command and view the registers. The bottom line of the display should now show SUB AX,BX. Trace and observe the registers. The contents are now AX ____ and BX ____ .

4. How would negative numbers appear in the register display? _____ Trace and find out. What are the results? _____

Procedure 3: Arithmetic with bytes

1. Arithmetic so far has been with four hex-digit words. The 8088 can also do math with bytes. Recall that general-purpose registers can be divided into high and low bytes. For example, AX is divided into AH and AL byte registers.

2. The instruction ADD AH,AL means to add the contents of AL to the contents of AH and leave the results in AH. The op code for ADD AH,AL is $00C4_H$. Enter the two bytes starting at address 0100_H by typing -E 100 00 (SPACE BAR) C4. Display the registers and load AX with 0204_H.

3. Next, add 04 and 02 in AL and AH. Notice that the numbers were loaded in AX. There is no command for loading the high and low register individually, but this is not a problem since the two bytes can be loaded into the 16-bit register as a word. Just remember that AX is a word register, AH and AL (or any of the four general-purpose registers) are byte registers.

4. Display the registers and verify the numbers in the registers, the op code at the starting address and the IP. Trace and observe the results. AH _____ AL _____

Procedure 4: Multiply and divide

1. The machine op code to multiply AX and BX is $F7E3_H$. When two 16-bit numbers are multiplied, the result can be a 32-bit answer. The 32-bits are too long to fit into a 16-bit register, so the 8088 stores the answer in both the AX and DX registers. The high 16-bits are placed in DX, and the low 16-bits are placed in AX. The two register combination can be written as DX:AX.

2. Place $F7E3_H$ in memory starting at 0100_H. Place $4A5_H$ in AX and 100_H in BX. Display the registers. Notice that the MUL BX instruction does not mention the AX register. (The reason is that to multiply words the 8088 always multiplies the register named in the instruction by the AX register, then stores the answer in the DX:AX pair of registers.)

3. Multiplication in hex is similar to multiplication in decimal. To multiply by 100 in decimal, simply add two zeros to the right of the decimal number. To multiply by 100 in hex, do the same, adding two zeros to the right of the hex number.

   To multiply $4A5_H$ by 100_H, add two zeros to $4A5_H$. The result, $4A500_H$, is too long to fit in one register, so divide into two words, 0004_H and $A500_H$.

4. Trace the instruction and observe the register contents. The DX:AX pair of registers contain DX _____ AX _____ .

5. Try multiplying three or four sets of other numbers. _____

6. Now divide $4A522_H$ by 100_H and see if the original number $4A5_H$ comes back with a remainder. The divide instruction is F7F3. Place $F7_H$ in address 0100_H and $F3_H$ in address 0101_H. Load DX with 0004_H and AX with $A522_H$. BX should still contain 0100_H as the divisor.

7. Divide $4A522_H$ by 100_H on pencil and paper. Move the radix point two places; the result should be 4A5 and 22 remainder.

8. Display the registers. DIV BX should display for the op code. The 8088 will use AX for the answer and DX for the remainder in a word divide operation.

9. Trace the instruction and examine the registers. AX _____ DX _____

Procedure 5: Meet an interrupt

1. In BASIC, the GOSUB command was used to go out to a subroutine, execute the subroutine, then return to the main program. The INT instruction (Interrupt) asks DOS to do something in a manner similar to GOSUB. The interrupt for DOS functions is INT 21h; the machine op code is $CD21_H$. Place $CD21_H$ in memory starting at address 0100_H. Place 0200_H in AX and 01_H in DX.

2. Display the registers. Set IP to 0100_H. DEBUG's Go command will run a program similar to RUN in BASIC. Since there still is no way to halt within the program, type -G 102. The program will execute through address 0100_H and 0101_H and stop when the IP points to address 0102_H. What happened?
_____

3. How did the happy face get on the screen? The 02_H in the AH register told DOS to print an ASCII character. DOS will print the ASCII character named in DX. Use the ASCII table handout **(HO 1.1)** and display the following ASCII characters. Write the ASCII code by each character, then display it.
A__ X__ S__ x__ a__ 2__ 9__ !__ #__ and several others.

Procedure 6: Meet a graceful interrupt

1. Until now, Trace has been used to execute only one instruction in a program, or G followed by the line number to stop. There has been no in-program method of stopping the microprocessor after the one-line program was finished. Recall that the remainder of the user memory contains "garbage", not valid instructions. To see the results of trying to execute trash, set the IP back to 0100_H. Type G for Go and stand back. What happened? _____

2. While it didn't explode, the computer probably did some weird things. It may lock up so that even CTRL-ALT-DEL will not warm boot. If this happens, cold boot the machine and reload DEBUG.

3. Load the registers in the same way as when you displayed ASCII characters, 02_H in AH and the ASCII code of your favorite character in DX. Load addresses 0100_H and 0101_H with $CD21_H$.

4. Another interrupt, INT 20, tells DOS to bring everything to a graceful but screeching halt, do not do the next thing IP points to, but set IP back to 0100 and return control back to DOS or DEBUG. In other words, you now have a HALT instruction!

5. Load $CD20_H$ in addresses 0102_H and 0103_H. Display the registers. Verify that the registers contain your numbers and IP is pointing to 0100_H. Notice that your first op code (INT 21) is displayed, but the second op code is not.

6. In BASIC, LIST was used to view the program's source code. DEBUG's U command (Unassemble) is similar to LIST. Give the U command to see your two-line program followed by "garbage".

7. Now give the G command. No stopping address number is needed since INT 20 is used. Use INT 20 to end a program from now on.

8. What happened? _____

Procedure 7: MOV instructions

1. The microprocessor will add a number in one register to a number in a second register with the mnemonic ADD.

   ADD AX,BX is a two-byte hexadecimal instruction, $01D8_H$.
   $01D8_H$ adds the contents of register BX to the contents of register AX.

2. Type -A 100

   ```
   MOV AH,02
   MOV DL,2A
   INT 21
   INT 20
   ```

   (Hit Return twice to get the - prompt) Display the registers; load AX and DX with 0000_H. Give the U command to view memory contents.

3. Type -G. Did the program run? _____ DEBUG assembled the source code (words that you typed) into machine code. The AH and DL registers were loaded by the MOV instructions.

Procedure 8: Print a string

1. INT 21 function 02_H prints a single character. Function number 09_H tells DOS to print a string and stop when it finds a $.

2. Use E 200 to start entering numbers in address 200_H, away from our program code. Type E 200 Return; DEBUG will display the current contents of 200_H, then wait for the new two-digit number to be entered. Type the 03 SPACE BAR 20 SPACE BAR 48 SPACE BAR, etc., according to the following.

   ```
   03  20  48  65  6C  6C  6F  20
   45  6C  65  63  74  72  6F  6E
   69  63  73  20  53  74  75  64
   65  6E  74  21  20  01  24
   ```

   The last number, 24, is the ASCII code for a $ and tells DOS it has reached the end of the string.

3. Now type -A 100 and enter the following:

   ```
   MOV AH,09
   MOV DX,0200
   INT 21
   INT 20
   ```

4. The instruction MOV AH,09 is to move immediate (the next number in memory) 09_H from memory to register AH. MOV AH,09 tells DOS to print a string. The instruction MOV DX,0200 is move immediate 0200_H from memory to register DX. The string starting address, 0200_H, is stored in DX.

What does INT 21 Function 09_H do? _____

What does INT 21 Function 02_H do? _____

What does INT 20 do? _____

5. Execute the program. What message printed? _____

6. Use the DEBUG D (Dump memory) command to view the memory starting at address 0200_H. Type -D 200. The second number displayed is the contents of address 0201_H to $020F_H$ at the right. The next line starts at 0210_H through $021F_H$, etc. At the far right of the display the contents of each memory location is displayed as an ASCII character. Your message should be displayed.

   DEBUG can only display the "normal" typewriter style characters. ASCII codes for special characters are displayed as a period. Note that an ASCII character will display even if that hex number is not intended to be an ASCII character, so some of the text display may be garbled.

7. Dump memory starting at address 100_H. Your program op codes are here; they are not meant to be ASCII characters, so the text display will be garbled.

8. Unassemble starting at 100_H. The hex numbers will be interpreted correctly as op codes and operands. Unassemble starting at 200_H. The display will show op codes and operands, but remember that the hex numbers here are really ASCII characters.

Procedure 9: Name program and save to disk

1. Give the program a file name (such as WRITEIT.COM) by typing -N WRITEIT.COM. The .COM extender is used so DEBUG will assemble the program in memory into an executable program on disk.

2. To save a file to disk, a byte count is needed. Since the program starts at 0100_H, use U to find the end of our program. Find the string and count across the display to the 24 representing ASCII $ to mark the end of the string. The address of the 24 is $021E_H$. The next address that IP would point to is $021F_H$. Find the difference between $021F_H$ and 0100_H with DEBUG's H command.

3. The byte count is $011F_H$. Place it in the BX:CX pair of registers as 0000 $011F_H$, 0000_H in BX and $011F_H$ in CX. The byte count must always be placed with the high word in BX and the low word in CX to tell DEBUG the length of the program.

4. Once DEBUG has been told the name and byte count, type -W to write the file to disk.

5. Quit DEBUG, then view the disk DIRectory. Is your .COM file on the disk? ____ Since it is a .COM file, it is a stand-alone program and will run simply by typing its name. Type the file name. Did it run OK? ____

6. Start DEBUG. Write a program to print your name and a short message on the screen. Assemble it and run it from DEBUG. Show the instructor.

7. Name the file and write it to disk. Run the .COM file from disk for the instructor.

Procedure:	Evaluation 3	2	1	N
1. Add registers	—	—	—	—
2. Subtract registers	—	—	—	—
3. Arithmetic with bytes	—	—	—	—

4.	Multiply and divide	—	—	—	—
5.	Meet an interrupt	—	—	—	—
6.	Meet a graceful interrupt	—	—	—	—
7.	MOV instructions	—	—	—	—
8.	Print a string	—	—	—	—
9.	Name program and save to disk	—	—	—	—
Average of above evaluations*		—	—	—	—

*This average is the student's partial evaluation for competencies I4-6.

INSTRUCTOR'S KEY TO JS 1.1

Procedure 1

3. Register display:

```
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0100 OV UP EI PL NZ NA PE CY
3965:0100 01D8          ADD     AX,BX
```

Unassemble display:

```
-u
3965:0100 01D8          ADD     AX,BX
3965:0102 38E8          CMP     AL,CH
```

4. Register display:

```
-r
AX=1111 BX=2222 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0100 OV UP EI PL NZ NA PE CY
3965:0100 01D8          ADD     AX,BX
```

5. Yes

6. AX holds 3333; BX holds 2222. The new sum should be 5555. IP holds 0102.

```
-t
AX=3333 BX=2222 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0102 NV UP EI PL NZ NA PE NC
3965:0102 38E8          CMP     AL,CH
```

7. AX holds 5555. (The student should try a variety of numbers.)

8. Before Trace:

```
-r
AX=1111 BX=2222 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0100 NV UP EI PL NZ NA PE NC
3965:0100 01D8          ADD     AX,BX
```

After Trace:

```
-t
AX=3333 BX=2222 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0102 NV UP EI PL NZ NA PE NC
3965:0102 38E8          CMP     AL,CH
```

AX = 3333, BX = 2222

Procedure 2

1-3. Before Trace:

```
-r
AX=3333 BX=2222 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0100 NV UP EI PL NZ NA PE NC
3965:0100 29D8          SUB     AX,BX
```

After Trace:

```
-t
AX=1111 BX=2222 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0102 NV UP EI PL NZ NA PE NC
3965:0102 38E8          CMP     AL,CH
```

(The student must remember to set IP to 0100_H.)

After Trace is executed, the register contents are: AX = 3333, BX = 2222 and BX was subtracted from AX.

4. A negative number forces the MSB to 1. (The student should execute Trace again. Watch that the student doesn't forget to set IP to 0100_H after each Trace.)

```
-t
AX=EEEF BX=2222 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0102 NV UP EI NG NZ AC PO CY
3965:0102 38E8          CMP     AL,CH
```

When 2222 is subtracted from 1111, a negative number is held in AX as AX = $EEEF_H$.

Procedure 3

4. Before Trace is executed:

```
AX=0204 BX=2222 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0100  NV UP EI NG NZ AC PO CY
3965:0100 00C4        ADD    AH,AL
```

After Trace is executed:

```
AX=0604 BX=2222 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0102  NV UP EI PL NZ NA PE NC
3965:0102 38E8        CMP    AL,CH
```

AH = 06 AL = 04

Procedure 4

2. Display:

```
AX=04A5 BX=0100 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0100  NV UP EI PL NZ NA PE NC
3965:0100 F7E3        MUL    BX
```

4. DX = 0004, AX = A500. Display:

```
AX=A500 BX=0100 CX=0000 DX=0004 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3965 ES=3965 SS=3965 CS=3965 IP=0102  OV UP EI PL NZ NA PE CY
3965:0102 38E8        CMP    AL,CH
```

5. The student should multiply several numbers and observe the registers.

8. Display:

```
AX=A522 BX=0100 CX=0000 DX=0004 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3ADC ES=3ADC SS=3ADC CS=3ADC IP=0100  NV UP EI PL NZ NA PO NC
3ADC:0100 F7F3        DIV    BX
```

9. AX = 04A5, DX = 0922. Display:

```
-t
AX=04A5 BX=0100 CX=0000 DX=0022 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3ADC ES=3ADC SS=3ADC CS=3ADC IP=0102  NV UP EI NG NZ NA PO CY
3ADC:0102 3AE8        CMP    CH,AL
```

Procedure 5

2. Display:

```
-t
AX=0200 BX=0000 CX=0000 DX=0001 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3ADC ES=3ADC SS=3ADC CS=3ADC IP=0100  NV UP EI NG NZ NA PO CY
3ADC:0100 CD21        INT    21
```

A "happy face" is printed on the CRT screen.

3. The ASCII code for a happy face is 01_H.

Char. decimal	ASCII code	Hex
A	65	41
X	88	58
S	83	53
x	120	78
a	97	61
2	50	32
9	57	39
!	33	21
#	35	23

Procedure 6

1. A "happy face" is printed on the CRT screen, then the computer locks up because it is trying to execute the random garbage in memory as if the garbage were valid op codes.

3. Display:

```
AX=0200 BX=0000 CX=0000 DX=0001 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3ADC ES=3ADC SS=3ADC CS=3ADC IP=0100  NV UP EI PL NZ NA PO NC
3ADC:0100 CD21        INT    21
```

5. Display:

```
AX=0200 BX=0000 CX=0000 DX=0001 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3ADC ES=3ADC SS=3ADC CS=3ADC IP=0100  NV UP EI PL NZ NA PO NC
3ADC:0100 CD21        INT    21
```

6. Display:

```
-u
3ADC:0100 CD21        INT    21
3ADC:0102 CD20        INT    20
```

8. The character in DX is printed on the screen, then the program control is handed back to Debug. Debug displays the following message on the screen:

```
-g
(Your character prints here)
Program terminated normally
-
```

Procedure 7

2. Assemble:

```
-
-A100
3ADC:0100 MOV AH,02
3ADC:0102 MOV DL,2A
3ADC:0104 INT 21
3ADC:0106 INT 20
3ADC:0108
```

Load registers:

```
-RAX
AX 0200
:0000
-RDX
DX 0001
:0000
-
```

Display registers:

```
-R
AX=0200 BX=0000 CX=0000 DX=0001 SP=FFEE BP=0000
SI=0000 DI=0000
DS=3ADC ES=3ADC SS=3ADC CS=3ADC IP=0100  NV UP
EI PL NZ NA PO NC
3ADC:0100 B402        MOV    AH,02
-
```

Unassemble:

```
-u
3ADC:0100 B402        MOV    AH,02
3ADC:0102 B22A        MOV    DL,2A
3ADC:0104 CD21        INT    21
3ADC:0106 CD20        INT    20
```

3. Yes. Debug assembled the source code, the program ran and then terminated.

Procedure 8

2. The Debug entry will look similar to the following:

```
-
-e200
3ADC:0200 00.03  00.20  00.48  00.65  00.6C  00.6C  00.6F  00.20
3ADC:0208 00.45  00.6C  00.65  00.63  00.74  00.72  00.6F  00.6E
3ADC:0210 00.69  00.63  00.73  00.20  00.53  00.74  00.75  00.64
3ADC:0218 00.65  00.6E  00.74  00.21  00.20  00.01  00.24
-
```

3. The Debug entry will look similar to the following:

```
-A100
3ADC:0100 MOV AH,09
3ADC:0102 MOV DX,0200
3ADC:0105 INT 21
3ADC:0107 INT 20
3ADC:0109
-
```

4. INT 21 Function 09_H tells DOS to print a string of characters and then stop printing when a $ is found.
INT 22 Function 02_H tells DOS to print a single character.
INT 20 exits the program and returns control to Debug.

5. Display:

```
-g
(heart) Hello Electronics Student! (happy face)
Program terminated normally
-
```

6. Display:

```
-d200
3ADC:0200  03 20 48 65 6C 6C 6F 20-45 6C 65 63 74 72 6F 6E   . Hello Electron
3ADC:0210  69 63 73 20 53 74 75 64-65 6E 74 21 20 01 24 00   ics Student! .$.
3ADC:0220  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
3ADC:0230  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
3ADC:0240  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
3ADC:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
3ADC:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
3ADC:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

7. Display:

```
-d100
3ADC:0100  B4 09 BA 00 02 CD 21 CD-20 C0 05 10 00 8B 0E 1E   ......!. .......
3ADC:0110  01 03 C8 89 4F FB 8B 0E-26 01 03 C8 89 4F F7 8B   ....O...&....O..
```

8. Display:

```
-u100
3ADC:0100 B409            MOV     AH,09
3ADC:0102 BA0002          MOV     DX,0200
3ADC:0105 CD21            INT     21
3ADC:0107 CD20            INT     20

-u200
3ADC:0200 0320            ADD     SP,[BX+SI]
3ADC:0202 48              DEC     AX
3ADC:0203 65              DB      65
3ADC:0204 6C              DB      6C
3ADC:0205 6C              DB      6C
3ADC:0206 6F              DB      6F
3ADC:0207 20456C          AND     [DI+6C],AL
3ADC:020A 65              DB      65
3ADC:020B 63              DB      63
3ADC:020C 7472            JZ      0280
3ADC:020E 6F              DB      6F
3ADC:020F 6E              DB      6E
3ADC:0210 69              DB      69
3ADC:0211 63              DB      63
3ADC:0212 7320            JNB     0234
3ADC:0214 53              PUSH    BX
3ADC:0215 7475            JZ      028C
3ADC:0217 64              DB      64
3ADC:0218 65              DB      65
3ADC:0219 6E              DB      6E
3ADC:021A 7421            JZ      023D
3ADC:021C 2001            AND     [BX+DI],AL
3ADC:021E 2400            AND     AL,00
```

Procedure 9

2. View the last part of the Unassemble display:

```
3ADC:021C 2001
3ADC:021E 2400 <— Address 021F_H
3ADC:0220 0000
```

or view the last part of the Dump display:

```
3ADC:0200  03 20 48 65 6C 6C 6F 20-45 6C 65 63 74 72 6F 6E   . Hello Electron
3ADC:0210  69 63 73 20 53 74 75 64-65 6E 74 21 20 01 24 00   ics Student! .$.
3ADC:0220  00 00 00 00 00 00 00 00-
                    Address 021F_H ———/
-h 021F 0100
031F  011F     <—  Difference is the byte count.
-
```

6. (To be evaluated by instructor)

7. The student must find the byte count of his/her program to save the program to disk.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT IV: MICROPROCESSOR INSTRUCTIONS

LESSON 2: DECISION-MAKING PROGRAMMING

NOTE: Upon completion of this lesson, the student should be able to test the flags register and program loops and jumps.

I. Terms and definitions

A. Jump--program flow where the IP is changed to go over part of the program and begin executing at a new location

B. Loop--program flow where the IP is changed to go back and repeat part of the program

C. Straight-line program--The program flow is in a straight line from one memory address to the next sequential address.

II. Flags

A. The flags register

1. The flags, or condition codes, register is examined as individual bits instead of a byte or word group of bits.

2. Flags can be considered as one-bit registers grouped into one 16-bit register.

3. There are nine flags, so only nine of the 16 bits are used.

4. Flags are divided into two groups.

a. Status

b. Control

B. Status flags

1. The chart below shows the bit position and name of the six status flags and the conditions shown.

Bit position	Name	
0	CF	Carry flag
2	PF	Parity flag
4	AF	Auxiliary carry flag
6	ZF	Zero flag
7	SF	Sign flag
11	OF	Overflow flag

2. Status flags hold the result of one instruction.

3. A program can check the status flag to find out what happened as a result of the previous instruction.

4. Only the CF (carry flag) can be easily modified in a program.

5. Status flags are primarily for observation, not manipulation by the program.

C. Control flags

1. The chart below shows the bit position and name of the three control flags.

Bit position	Name	
8	TF	Trap flag
9	IF	Interrupt enable flag
10	DF	Direction flag

2. Control flags are manipulated by a program to determine how the microprocessor will respond to coming instructions.

D. Condition of flags shown by DEBUG

1. DEBUG's R command displays the condition of the flags as mnemonics on the right of the middle display row.

2. DEBUG does not display the TF (trap) flag. It is best to leave TF alone unless you want to "crash" the system.

3. A mnemonic is displayed by DEBUG when the flag is set or reset.

	Mnemonic display when	
Status flag name	Set	Reset
CF (did a carry occur?)	CY (yes)	NC (no)
PF (was parity even/odd?)	PE (even)	PO (odd)
AF (did auxiliary carry occur?)	AC (yes)	NA (no)
ZF (was result a zero?)	ZR (yes)	NZ (no)
SF (if signed number)	NG (negative)	PL (positive)
OF (did overflow occur?)	OV (yes)	NV (no)
Control flag name	Set	Reset
IF (is interrupt?)	EI (enabled)	DI (disabled)
DF (did it INC or DEC?)	DN (decrement)	UP (increment)

E. Rules for zero flag

1. During every instruction that performs arithmetic, the 8086/88 will set or clear the Z flag.

2. If a calculation results in 0, ZF will be set and will stay set until a calculation occurs that results in a non-zero.

3. If a calculation results in other than zero, ZF will clear (reset) and stay cleared until a calculation occurs that results in zero.

4. ZF always shows whether the last calculation resulted in a zero or non-zero.

F. Rules for sign flag

1. When the most significant bit (MSB) of a result is 1, SF is set. If the result is 0, SF is reset.

2. If the result is a binary word, bit 15 determines the condition of the flag.

3. If the result is a binary byte, bit 7 determines the condition of the flag.

4. SF is meaningful only with signed numbers where the MSB indicates a positive or negative number.

5. SF is meaningless on unsigned numbers.

G. Rules for carry flag

1. If FF_H is added to FF_H, will eight bits hold the result?

```
  1111 1111        (FF_H)
+ 1111 1111      + (FF_H)
1 1111 1110      (1FE_H)
```

2. The sum results in nine bits; the register only holds eight bits!

3. If $FFFF_H$ is added to $FFFF_H$, the following results.

```
  1111 1111   1111 1111     (FFFF_H)
+ 1111 1111   1111 1111   + (FFFF_H)
1 1111 1110   1111 1111     (1FFFE_H)
```

4. The sum results in 17 bits, but the word register only holds 16 bits. (If you depended on the register while calculating your paycheck, you would lose part of your pay.)

5. The carry flag (CF) will hold the extra bit.

6. The program can examine CF to see if a carry occurred.

   a. If a carry occurred, CF is set.

   b. If a carry did not occur, CF is clear.

7. When subtracting a large number from a smaller number, a borrow must occur.

8. In this case, CF will act as a borrow flag.

   a. If CF is set, a borrow occurred.

   b. If CF is clear, a borrow did not occur.

9. If a larger number is subtracted from a smaller number, CF is set.

H. Change flags with DEBUG

1. Flags can be changed in DEBUG with the command RF.

2. DEBUG will display the condition of the flags and wait for the mnemonic of the change to be entered.

3. For example, if NG were displayed (SF indicating a negative number), enter PL to reset the flag for a positive number.

I. Change flags with a program

1. The 8086/88 has seven instructions for manipulating flags.

2. See the chart below for the assembly language mnemonic, hex op codes and the operation performed.

Mnemonic	Op code	Operation
CLD	FC	Clear (reset) direction flag
STD	FD	Set direction flag
CLI	FA	Clear (reset) interrupt enable flag
STI	FB	Set interrupt enable flag
CLC	F8	Clear (reset) carry flag
CMC	F5	Complement carry flag
STC	F9	Set carry flag

III. Decision-making

A. Loops

1. Programming until now has involved basic instructions for the 8086/88 and writing simple straight-line programs to examine the operation of the instructions and registers.

2. An exception was the DB pseudo-op where the pseudo-op was placed someplace else in memory and the memory was told where to find it.

3. Of course, INT 21 function 09 and 02 did branch off to subroutines provided by DOS.

4. The interrupts, though, were placed in our straight-line program in the proper sequence of execution.

5. The real power of a computer is the ability to test, make a decision, then go to another set of instructions someplace in memory based on the decision.

6. The ability to test-and-loop or test-and-jump is an important part of any computer language.

7. It is so important that the 8086/88 has several op codes devoted to the ability to test and make a decision.

8. A decision loop takes the following general form.

```
Initialize a counter          (Set up a counter)

Start of loop                 (Body of the loop)
                              (Starts here)
Instructions                  (Instructions to be)
                              (repeated go in body of)
                              (the loop)

Decrement the counter         (Count down by 1)
                              (Each pass through the loop)

If counter < >0               (Test for 0)
goto start of loop            (and make a decision to)
or                            (either loop back or drop)
                              (out of loop)
If counter = 0                (and go on to)
fall out of loop              (to the next)
                              (instruction)
Next instructions

Loop format
```

9. First a counter is built by loading into a register the number of times the loop should repeat.

10. The operations to be done are instructions in the body of the loop. After the operations are done, several steps occur.

    a. The counter is incremented to indicate one pass has been completed through the loop.

    b. ZF is conditioned by the decrement.

    c. ZF is tested to see if the counter has reached zero. If not zero, the decision is made to loop back to the start of the loop and repeat.

    d. When the counter does reach zero, the decision is made to fall through the loop to the next memory location and execute the instruction found there.

11. This format is similar to If-Then and GoTo in BASIC.

B. Conditional jump **(HO 2.1)**

1. JNZ (jump if not zero) is one of the 8086/88's conditional jump instructions.

2. Conditional jumps are two bytes long.

   Example: Op code byte 75_H Displacement byte $0F_H$

3. The high-order byte tells the 8086/88 which status flag to test and for what value.

   Examples:

Mnem.	Op code	Operation
JO	70	Jump if O flag = 1 (if overflow)
JNO	71	Jump if O flag = 0 (if not overflow)
JC	72	Jump if C flag = 1 (if carry occurred)
JNC	73	Jump if C flag = 0 (if no carry)
JZ	74	Jump if Z flag = 1 (if zero)
JNZ	75	Jump if Z flag = 0 (if not zero)

4. The low-order byte is a displacement byte that tells the microprocessor where to go if the test is "true" (not equal to zero).

5. Recall that the IP (instruction pointer) always points to the next instruction to be executed.

6. If the condition tested is met (ZF clear), the contents of the displacement byte is added to the contents of IP to point to the new address and instruction to be executed.

7. If the condition test is not met (ZF set), IP advances to the next memory address and falls out of the loop.

8. The conditional jump can be forward or backward.

9. If the MSB of the displacement byte is a 0 (clear), the jump is forward.

10. If the MSB of the displacement byte is a 1, MSB is set and the jump is backward.

11. Since only eight bits are being used in the displacement byte, there are 256 number combinations.

12. The range of the jump is limited to 127 bytes forward and 128 bytes backward.

13. Examine the following listing.

```
0100 MOV DX,05  Load a constant into DX
0103 MOV AX,00  Clear AX
0106 INC AX     Increment AX by 1
0107 DEC DX     Decrement DX by 1 and condition ZF
0108 JNZ 106    If Z flag not set, loop back to 0106
010A INT 20     Halt
```

14. A constant is loaded into DX to initialize a counter by the first instruction.

15. The next instruction (clear AX to 0) and the increment AX instruction are not really a part of this demonstration. They are simply something to do and are a counter to tell how many times the loop has executed.

16. If the Z flag was not already conditioned to reset, the increment instruction will do so since it is an arithmetic operation.

17. The decrement DX instruction is also an arithmetic operation and will affect the Z flag.

18. The JNZ instruction immediately follows the decrement so that the JNZ will read the ZF condition caused by our counter register DX.

19. If the Z flag is not set, a decision to loop back to 0106 takes place.

20. The IP is now pointing to memory address 010A, but address 0106 is jammed into the IP forcing the loop back.

21. Each loop will add one to AX and subtract one from DX.

22. When DX is finally decremented to 0, the Z flag is conditioned to 1 and program execution is allowed to proceed to 010A as the loop "falls through".

23. Jumps are so important that the 8086/88 has 17 different conditional jumps, as shown in **HO 2.1.**

24. The instructions in Group 2 **(HO 2.1)** are tests for combinations of flags and are generally used after a CMP (compare) instruction.

25. JCXZ in Group 3 does not test flags but checks to see if register CX is zero.

C. Unconditional jump

1. There are times the programmer will want to change the IP without doing a flag or register test.

2. The unconditional jump mnemonic, JMP, performs as a Jump Always instruction.

3. There are five distinct jump instructions with the JMP mnemonic.

4. One form of the unconditional jump is like the conditional jump but without a test.

5. This form is op code EB_H in the first byte and a signed displacement byte that is added to the IP.

6. This is a relative jump.
    a. The displacement byte is not the absolute address or destination.
    b. The displacement byte is added to the IP to get the destination.
7. The jump can be -128 bytes or +127 bytes.
8. If the displacement were two bytes, there would be 64K possible bytes in the jump range.
9. This is a three-byte form with op code $E8_H$ in the first byte.
10. The two displacement bytes are the relative address added to the IP to get the absolute address.
11. The new IP value is calculated without a sign.
12. The two displacement bytes allow a jump anywhere within the 64K code segment.

D. Near jumps

1. Most jumps are within the range of the two-byte short form.
2. Two bytes take less memory and are slightly faster than the three-byte long form.
3. An assembler makes the decision for the operator using the short form where possible and the long form where more range is needed.
4. Recall that a conditional jump has a range of -128 and +127.
5. A JMP instruction can be used to extend the jump range of the conditional jump.
6. Jumps within the code segment are called near jumps.

E. Far jumps

1. Jumps made between code segments are far jumps.
2. The five-byte far jump has an op code byte, EA_H and four data bytes.
3. The far jump does not use relative addressing but specifies both IP and CS values.
4. The least significant part of the address (the IP value) is stored in low memory.
5. The most significant part of the address (the CS value) is stored in high memory.
6. Jumps are used to change the instruction pointer and program control to a different place in the program.

F. Loop

1. Loops are so important in computer programs that the 8086/88 has a set of loop instructions.
2. The mnemonic LOOP is followed by a displacement byte.
3. Recall that the primary purpose of CX is as a count register.

4. LOOP decrements CX by 1 and transfers control to the target operand if CX is not zero.

5. If CX is zero, the loop is not taken and the IP points to the following instruction.

G. A "do nothing" instruction

1. In BASIC, lines of instruction are usually numbered by 10s so that there is room to go back and add instructions if needed.

2. There is no way to add something "between the lines" when using DEBUG's mini-assembler.

3. This is not critical on short programs, but imagine a program listing of several hundred instructions.

4. To add an instruction or change an instruction with a different number of bytes, enter all the remaining instructions.

5. The NOP (no operation) instruction is a do-nothing, one-byte long instruction that simply kills a little time.

6. The NOP can be included throughout a program to give a little extra room if a change is needed in the listing.

7. NOPs can be removed as needed to correct the new listing without typing the whole listing over.

8. The NOP uses three clock cycles while executing.

9. If loops are used for timing purposes, use the NOP to fine tune the timing loop.

CONDITIONAL JUMP INSTRUCTIONS.

Mnemonic	Op code	Test and operation
Group 1		
JO	70	JUMP IF O = 1 (if overflow)
JNO	71	JUMP IF O = 0 (if not overflow)
JC	72	JUMP IF C = 1 (if carry)
JNC	73	JUMP IF C = 0 (if not carry)
JZ	74	JUMP IF Z = 1 (if zero)
JNZ	75	JUMP IF Z = 0 (if not zero)
JS	78	JUMP IF S = 1 (if sign)
JNS	79	JUMP IF S = 0 (if not sign)
JP	7A	JUMP IF P = 1 (if parity)
JNP	7B	JUMP IF P = 0 (if not parity)
Group 2		
JBE	76	JUMP IF (C = 1) OR (Z = 1)
JA	77	JUMP IF (C = 0) AND (Z = 0)
JL	7C	JUMP IF S <> 0
JGE	7D	JUMP IF S = 0
JLE	7E	JUMP IF ((S XOR O) OR Z) = 1
JG	7F	JUMP IF ((S XOR O) OR Z) = 0
Group 3		
JCXZ	E3	JUMP IF CX = 0

DEMONSTRATE FLAGS AND LOOPS

NOTE: After completing this experiment the student will be able to use conditional and unconditional jumps, test flags and registers, use the XCHG mnemonic, program loops and analyze program flow.

Equipment:

IBM PC or compatible
DOS diskette with DEBUG
File diskettes

Procedure 1: Flags and conditional jumps

Evaluation 3 2 1 N

1. Assemble the following listing. — — — —

```
0100  MOV  DX,0005
      MOV  AX,0000
      NOP
      INC  AX
      DEC  DX
      JNZ  0103
```

2. Unassemble and observe the op codes. Transfer information to Table 1. Show the loop with arrows. — — — —

Table 1

Address	Op code	Operand	Comments
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

3. Trace through the program. Observe the condition of the Z flag and the contents of IP, AX and DX. Place your observed data in Table 2. Show where the program falls through the loop. — — — —

Table 2

Pass #	ZF (mnemonic)	ZF (binary)	AX	DX	IP
1					
2	—	—	—	—	—
3	—	—	—	—	—
4	—	—	—	—	—
5	—	—	—	—	—
6	—	—	—	—	—
7	—	—	—	—	—
	—	—	—	—	—

4. Describe the operation of the program. — — — —

Procedure 2: XCHG

1. Write your own program to load a number into registers CX, DX, BH, DH, BP, SI and AX. Use a different number for each register. Add the following instructions to the program to swap the registers. — — — —

```
____    _____
____    _____
____    _____
____    _____
____    _____
____    _____
____    _____

XCHG    CX,DX
XCHG    BH,DH
XCHG    BP,SI
XCHG    SI,AX
XCHG    DX,AX
____    _____
```

2. Assemble the program then unassemble to fill Table 3 with the program information. — — — —

Table 3

Address	Op code	Operand	Comments
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

3. Make your own Table 4 similar to Table 2 using Trace and showing register contents and IP at each single step. — — — —

Table 4

_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

4. Describe the operation of the program.

Procedure 3: Loop

1. Assemble the following code. Add your own instructions at the end, using the DB pseudo-op to print the message "Loop the Loop" when the program falls out of the loop.

```
0100 MOV      CX,0006
     JCXZ 0109
     NOP
     NOP
     LOOP0105
```

2. Enter the information for your completed program in Table 5.

Table 5

Address	Op code	Operand	Comments

3. Make your own Table 6 similar to Table 2 showing the contents of registers and IP at each single step (Trace).

Table 6

4. Describe the operation of the program.

Procedure 4: Unconditional jumps

1. Assemble the following program. _ _ _ _

```
0100  JMP     0109
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      JMP     07F0
      JMP     F000:F065
```

2. Complete the following table for this program. _ _ _ _

Table 7

Address	Op code	Operand	Kind of jump? Comments

3. Make your own Table 8 similar to Table 2 showing the contents of registers and IP at each single step (Trace). _ _ _ _

Table 8

4. Describe the operation of the program. _ _ _ _

5. Turn off computer power and store diskettes. _ _ _ _

Average of above evaluations* _ _ _ _

*This average is the student's partial evaluation for competencies I4-6.

160

INSTRUCTOR'S KEY TO JS 2.1

Procedure 1

2. Display:

```
   0100 BA0500        Move 05 into register DX
-->0103 B80000        Move 0000 into register AX
|  0106 90            no operation
|  0107 40            Increment AX by 1
|  0108 4A            Decrement DX by 1
|-- 0109 75F8         If not 0 jump to address 0103
```

3. The student will trace each op code of the program. Notice that just before DX is decremented IP points to 0109. When DX is decremented but is not zero, flag ZF remains reset. (A binary 0 indicated by ZF display on flags) The jump instruction tests the flag, finds it to be reset, then loads I.P. with address 0103 for the jump. When the decrement results in zero, flag ZF sets (indicated by ZR on the flag display), falls out of the loop, and the CPU tries to execute whatever is found at address 0109.

Table 2

Pass #	ZF	ZF	AX	DX	I.P.
1	NZ	0	0001	0004	0103
2	NZ	0	0001	0003	0103
3	NZ	0	0001	0002	0103
4	NZ	0	0001	0001	0103
5	ZR	1	0001	0000	010B (or past 0109)

4. See description in Step 3 above.

Procedure 2

1. The student will load numbers into registers. For example:

```
0100 MOV CX,1111
0103 MOV DX,2222
0106 MOV BH,33
0108 MOV DH,44
010A MOV BP,5555
010D MOV SI,6666
0110 MOV AX,7777
0113 XCHG CX,DX
0115 XCHG BH,DH
0117 XCHG BP,SI
0119 XCHG SI,AX
011A XCHG DX,AX
011B INT 20
```

2. With the example program unassembled, use information to complete Table 3.

```
3ADC:0100 B91111      MOV     CX,1111
3ADC:0103 BA2222      MOV     DX,2222
3ADC:0106 B733        MOV     BH,33
3ADC:0108 B644        MOV     DH,44
3ADC:010A BD5555      MOV     BP,5555
3ADC:010D BE6666      MOV     SI,6666
3ADC:0110 B87777      MOV     AX,7777
3ADC:0113 87CA        XCHG    CX,DX
3ADC:0115 86FE        XCHG    BH,DH
3ADC:0117 87EE        XCHG    BP,SI
3ADC:0119 96          XCHG    SI,AX
3ADC:011A 92          XCHG    DX,AX
3ADC:011B CD20        INT     20
```

Table 3

Addr.	Op Code	Operand	Comments
0100	B9	1111	Move 1111 to CX
0103	BA	2222	Move 2222 to DX
0106	B7	33	Move 33 to BH
0108	B6	44	Move 44 to DH
010A	BD	5555	Move 5555 to BP
010D	BE	6666	Move 6666 to SI
0110	B8	7777	Move 7777 to AX
0113	87CA		Exchange contents of CX and DX
0115	86FE		Exchange contents of BH and DH
0117	87EE		Exchange contents of BP and SI
0119	96		Exchange contents of SI and AX
011A	92		Exchange contents of DX and AX
011B	CD20		Exit program, return to Debug

3. Table 4

Trace	CX	DX	BH	DH	BP	SI	AX	I.P.
1	1111	0000	00	00	0000	0000	0000	0103
2	1111	2222	00	22	0000	0000	0000	0106
3	1111	2222	33	22	0000	0000	0000	0108
4	1111	4422	33	44	0000	0000	0000	010A
5	1111	4422	33	44	5555	0000	0000	010D
6	1111	4422	33	44	5555	6666	0000	0110
7	1111	4422	33	44	5555	6666	7777	0113
8	4422	3311	11	33	5555	6666	7777	0117
9	4422	3311	11	33	6666	5555	7777	0119
10	4422	3311	11	33	6666	7777	5555	011A
11	4422	5555	11	55	6666	7777	3311	011B

4. Each register was loaded with a number by the MOV instructions, then contents of registers were swapped with the XCHG instructions.

Procedure 3

1. Program listing

```
-a100
3ADC:0100 MOV CX,0006
3ADC:0103 JCXZ 0109
3ADC:0105 NOP
3ADC:0106 NOP
3ADC:0107 LOOP 0105
3ADC:0109
-
```

2. Table 5

Address	Op code	Operand	Your comments
0100	B9	06	Move 06 into CX
0103	E3	04	When CX is zero jump to address 0109
0105	90		NOP
0106	90		NOP
0107	E2	FC	LOOP back four addresses to 0105

Trace#	CX	I.P.
1	0006	0103
2	0006	0105
3	0006	0106
4	0006	0107
5	0005	0105
6	0005	0106
7	0005	0107
8	0004	0105
9	0004	0106
10	0004	0107
11	0003	0105
12	0003	0106
13	0003	0107
14	0002	0105

15	0002	0106
16	0002	0107
17	0001	0105
18	0001	0106
19	0001	0107
20	0000	0109

4. CX is loaded with the loop count number. The second instruction tells the CPU "when CX is 0, fall out of the loop and point I.P. to address 0109." Each time the loop is executed, CX is decremented by 1. If CX is not zero, I.P. is changed to loop back to address 0105. The NOP instructions are just fillers. When CX is zero, the I.P. points to address 0109 and falls out of loop.

Procedure 4

2.

Table 7

Address	Op code	Operand	Comments
0100	EB	07	Jump to address 0109
0102	90		NOP
0103	90		NOP
0104	90		NOP
0105	90		NOP
0106	90		NOP
0107	90		NOP
0108	90		NOP
0109	E9	E406	Jump to address 07F0
010C	EA	65F000F0	JMP F000:F065

3. Instructor to evaluate based on the following.

Trace#	I.P.
Start	0100
1	0109
2	07F0

4. The first instruction is an unconditional jump. Contents of the I.P. is added to 07; I.P. points to address 0109. CPU jumps over all of the NOP instructions. Address 0109 has an unconditional jump to address 07F0. The instruction at 010C is never executed as the CPU has no way to get back to this instruction.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT IV: MICROPROCESSOR INSTRUCTIONS

LESSON 3: ADDRESS MODES

NOTE: This lesson introduces the use of register address mode, immediate mode, direct and indirect modes of addressing. Use the DB pseudo-op and output data to the printer port.

I. Definitions

A. Op code--mnemonic converted to a binary number that tells the microprocessor what to do

B. Pseudo-op--false op code that tells the assembler what to do

C. Toggle--turn off and on; binary switching from high to low to high; similar to a switch being flipped off and on

II. Addressing modes

A. Register address mode

1. In the register mode, the operand resides in one of the internal registers of the microprocessor.

   Example: MOV BX,AX

2. The instruction will move the contents of register AX (source operand) to register BX (destination operand).

3. Register mode will move only operands between registers.

B. Immediate address mode

1. In the immediate mode, the operand is part of the instruction, not a register or memory location.

2. Immediate operands are usually constants.

3. For example, MOV BL,$01B_H$ moves the constant $01B_H$ into register BL.

4. The move immediate instruction tells the microprocessor to move the operand found in the memory location immediately following the instruction into register BL.

C. Direct address mode

1. In direct mode, the memory locations following the op code contain the address where the operand can be found, not the actual operand.

2. MOV AX,0250 tells the microprocessor that the address following the op code contains a 16-bit address.

3. The address is the location of the actual data to be moved into AX.

4. The location following the op code contains an effective memory address (EA).

5. EA is an offset for the current value in the data segment (DS) register.

6. In the BIU, the EA is added to the contents of DS to find the physical address of the operand.

7. The offset is also known as the displacement.

D. Indirect address mode

1. If the instruction MOV DL,200 is implemented, the operand 0200_H is moved to register DL.

2. Instead, if MOV DL,[200] is written, the contents of memory location 0200_H are placed in DL.

3. Consider the following listing.

```
0100 MOV DL,[200]
0104 MOV AH,5
0106 INT 21
0108 MOV DL,[201]
010C MOV AH,5
010E INT 21
0110 MOV DL,[202]
```

4. At line 0100 the contents of memory location 0200_H is moved to DL, then printed.

5. At line 0108 the content of the next memory location, 201, is moved to DL and then printed.

6. Then the procedure is repeated again at line 0110.

E. Register indirect address mode

1. The EA resides in either an index register or a pointer register.

2. The index register can be either source index register (SI) or destination index register (DI).

3. The pointer register can be either base register (BX) or base pointer register (BP).

   Example: MOV DL,[BX] takes the number found in register BX and adds it to the number found in register DS to find the physical address.

4. The operand found at this physical address is then moved to register DL.

5. To further clarify this concept, examine the following listing.

```
0100 MOV CX,xx
0103 MOV BX,11A
0106 MOV DL,[BX]
0108 MOV AH,5
010A INT 21
010C INC BX
010D LOOP 106
010F INT 20
011A DB 'Electronics Technology' 0D,0A
```

6. At line 0103_H, BX is loaded with the starting address of a text string.

7. Register BX has a special property.

   a. To put a memory address in BX, use BX with square brackets around it in an instruction.

   b. Any reference to [BX] will operate on the contents of the memory address residing in BX, not on BX itself.

8. BX now contains a pointer to address $011A_H$.

9. The next instruction does not load the contents of BX into DL, but instead points to address $011A_H$ and loads the contents of that address (ASCII code for E) into DL.

10. Instead of writing an instruction to get the next text character in the string, increment BX.

11. When BX is incremented at line $010C_H$, the pointer is incremented and points to the next address of the text string.

12. When the program loops back, the next text character is printed.

13. By incrementing BX and then looping, the complete text string will be printed one character at a time.

TERMS AND DEFINITIONS

Match the following programming terms and definitions.

____	1.	Base 16 number system using numbers 0-9 and letters A-F	a. Hexadecimal
____	2.	Turn off and on; binary switching from high to low to high	b. Interrupt
____	3.	Function call asking DOS to use a built-in assembly language routine to do something for the programmer; saves the programmer time	c. Jump d. Loop
____	4.	Program flow where the IP is changed to go back and repeat part of the program	e. Op code ab. Pseudo-op
____	5.	Program flow where the IP is changed to go over part of the program and begin executing at a new location	ac. Radix point ad. Straight-line
____	6.	Mnemonic converted to a binary number that tells the microprocessor what to do	ae. Toggle
____	7.	False code that tells the assembler what to do	

ANSWERS TO AS 3.1

1. a
2. ae
3. b
4. d
5. c
6. e
7. ab

USE ADDRESSING MODES

NOTE: On this job sheet the student will use INT 21, function 0A (keyboard buffer); INT 21, function 05 (printer output); indirect memory addressing mode, indirect addressing with a register pointer and DB pseudo-op.

Equipment:

IBM-PC compatible computer
MS-DOS or PC-DOS
DEBUG

Procedure 1: Buffered keyboard input

Evaluation 3 2 1 N

1. PC-DOS INT 21 Function $0A_H$ sets up a buffer area in memory to hold keystrokes. Assemble the following listing. — — — —

```
-A100
0100 MOV DX,109
0103 MOV AH,A
0105 INT 21
0107 INT 20
0109 DB 20
```

2. DX holds the address of DB 20. DB 20_H tells DEBUG to set aside 20 Hex, (32 decimal) unused memory locations as a buffer or temporary storage for strokes read from the keyboard. — — — —

3. AH holds keyboard buffer function, A or 0A, and the INT 21 is called. INT 20 is the usual exit routine. — — — —

4. Unassemble the program. Your display should be similar to the following. — — — —

```
0100 BA0901      MOV  DX,0109
0103 B40A  MOV   AH,0A
0105 CD21  INT   21
0107 CD20  INT   20
0109 201F        AND  [BX],BL
```

NOTE: Recall that the U command interprets numbers in memory as op codes. The pseudo-op at address 0109 has told DEBUG to reserve memory locations for ASCII codes. Address 109, then, will have to be examined for ASCII codes with the D command. To dump one address only, type -D 109,109 for starting and ending of the dump at address 0109.

5. Dump 109 to verify that 20 locations are reserved. Your display is _____ _____. — — — —

6. The buffer is a sequence of memory locations. The first byte of the buffer will hold the number that is specified (20), which is the maximum number of characters the buffer will hold. The second byte will hold the actual number of characters stored in the buffer. The remaining bytes will hold the actual ASCII codes for the characters typed. — — — —

7. Type G to run the program. Type a sentence from the keyboard. If you try to exceed the buffer length, the PC will beep and stop entering keystrokes. Hit return and the program will terminate. Now dump memory starting at address 0109 and observe the buffer area. List the contents of the buffer below. _____ _____ _____ _____

Table 1: Buffer contents

Address	ASCII	Char.	Address	ASCII	Char.

8. Find the first two bytes of the buffer and mark what they show. Place the character next to each ASCII code of your message. _____ _____ _____ _____

9. Repeating several times, use a shorter message at least one time and dump the buffer contents below. Do remnants of a longer message remain in the buffer when a short message is typed? ___ _____ _____ _____ _____

What character flags the end of the message string? _____

Table 2: Buffer contents

Address	ASCII	Char.	Address	ASCII	Char.

Procedure 2: Echo program

1. Next, put together two DOS function calls to make a program that takes a sentence that is typed in, stores it in a buffer, then echoes it back onto the screen. Assemble the following program. — — — —

```
0100 MOV DX,116      (Buffered keyboard input routine)
0103 MOV AH,A
0105 INT 21

0107 MOV DL,A        (Display output routine..linefeed)
0109 MOV AH,2        (Line feed moves text up one line)
010B INT 21          (ASCII character 0A is line feed)

010D MOV DX,118      (Print string routine)
0110 MOV AH,9
0112 INT 21

0114 INT 20          (Return to DEBUG or DOS)

0116 DB 30           (Maximum number of characters)
```

2. Complete Table 3. Run the program. Type in a sentence and watch it echo back. — — — —

Table 3

Address	Op code	Operand	Comments

_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

3. Did it display your sentence and then "garbage"? _____ How can the sentence be terminated so garbage will not be displayed? — — — —

_____

Try it. Also try changing the number of characters in the buffer. Since one byte holds the maximum number of characters the buffer will hold, just what is the maximum number a buffer can hold? _____

4. When the echo program is working properly, list and label your memory dump in Table 4. — — — —

Table 4: Buffer contents

Address	ASCII	Char.	Address	ASCII	Char.
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____
_____	_____	_____	_____	_____	_____

Procedure 3: Send character to printer

1. INT 21 Function 05_H will output text to the printer one character at a time. The ASCII code for the character must be moved to the DL register. Use the DB pseudo-op to assemble the ASCII codes into memory, then use indirect addressing to locate the ASCII codes and move them to the DL register. — — — —

2. The printer will not print anything until a full line of 80 characters is received or until a carriage return is received. The ASCII code for carriage return is $0D_H$. When the carriage return is received, the print head will move back to the left side of the paper. If another text string is sent to the printer, it will be printed over the first line. To prevent overprinting a line, a line feed must be sent to the printer to tell it to roll the paper up one line for the next line of text. The ASCII code for line feed is $0A_H$. — — — —

3. To send a message shorter than 80 characters to the printer ("hello", for example), send "hello", carriage return and line feed, so the message will be printed, the print head returned and the paper rolled up to the next line. This requirement will be met if you place DB 'hello',0D,0A at memory location 0200_H. The DB pseudo-op will tell DEBUG to place the text string in memory starting at 0200_H in the following manner. — — — —

0200 h
0201 e
0202 l
0203 l
0204 o
0205 carriage return
0206 line feed

4. Assemble the following program. — — — —

```
0100 MOV DL,[200]    'Move contents of 0200H to DL
0104 MOV AH,5        'Initialize printer out function
0106 INT 21          'Call DOS to Print it
0108 MOV DL,[201]    'Move contents of next memory location to DL
010C MOV AH,5        'Get ready to print
010E INT 21          'Do it again
0110 MOV DL,[202]    'Get next character
0114 MOV AH,5        'Get ready to print
0116 INT 21          'Print it
0118 MOV DL,[203]    'Get next character
011C MOV AH,5        'Get ready to print
011E INT 21          'Print it again
0120 MOV DL,[204]    'Get next character
0124 MOV AH,5        'Get ready to print
0126 INT 21          'Print it
0128 MOV DL,[205]    'Find the carriage return
012C MOV AH,5        'Get ready to send it
012E INT 21          'Print it (return print head)
0130 MOV DL,[206]    'Find the line feed
0134 MOV AH,5        'Get ready to send it
0136 INT 21          'Send and line feed
0138 INT 20          'Terminate program
```

5. Dump the text string and complete Table 5 below. — — — —

Table 5

Address	ASCII	Character
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

6. Save the program to disk--don't forget the ASCII text string in the byte count. Unassemble the program and then complete Table 6. Analyze the program and add your comments as to what each instruction will do. — — — —

Table 6

Address	Op code	Operand	Comments

9. What must occur before the instruction at address $010D_H$ will fall out of the loop? _____ — — — —

Why is BX loaded with $11A_H$? _____

What is loaded into DL? _____

Is it BX or the contents of BX?_____

Why is BX incremented?_____

Use the answers to these questions to help complete Table 7.

10. Save the program to disk, then analyze it thoroughly before running it. — — — —

Table 7

Address	ASCII or op code	Char. or operand	Comments

Average of above evaluations* _____

*This average is the student's partial evaluation for competencies I4-6.

INSTRUCTOR'S KEY TO JS 3.1

Procedure 1

4. Display:

```
-U100
3ADC:0100 BA0901        MOV     DX,0109
3ADC:0103 B40A          MOV     AH,0A
3ADC:0105 CD21          INT     21
3ADC:0107 CD20          INT     20
3ADC:0109 2005          AND     [DI],AL
```

5. Display:

```
-d109,109
3ADC:0100                               20
```

7-8. **Example**: If "Hello Student" is entered, the following will be observed. The student may use a different message.

```
-g
Hello Student
Program terminated normally
-
```

Memory dump starts at address 0109.

```
-D109
3ADC:0100                               20 0D 48 65 6C 6C 6F             .Hello
3ADC:0110  20 53 74 75 64 65 6E 74-0D 01 03 C8 89 4F F7 88    Student.....O..
```

Table 1

Addr.	ASCII	Character
0109	20	This byte holds number of bytes in buffer
010A	OD	This byte holds number of bytes in message
010B	48	H
010C	65	e
010D	6C	l
010E	6C	l
010F	6F	o
0110	20	SPACE
0111	53	S
0112	74	t
0113	75	u
0114	64	d
0115	65	e
0116	6E	n
0117	74	t
0118	0D	Flags end of string

In this message there are 13 characters; address 010A holds $0D_H$, or 13 in decimal.

9. Yes. Remnants of a longer message will remain in the buffer if they are not overwritten by a short message. Character 0D will always flag the end of a string in the buffer. Table 2 will depend on what the student has entered from the keyboard; it will be similar to Table 1. The first byte of the buffer holds number of bytes in buffer. The second byte of buffer holds number of actual characters. The character string always terminates with 0D.

Procedure 2

2. Display:

```
0100 BA  1601    Load DX with buffer address 0116
0103 B4  0A      Load AH with function 0A
0105 CD  21      Call DOS INT 21
0107 B2  0A      Load DL with line feed character 0A
0109 B4  02      Load AH with function 02
010B CD  21      Call DOS INT 21
010D BA  1801    Load DX with start of message address 0118
```

```
0110 B4  09      Load AH with function 09
0112 CD  21      Call DOS INT 21
0114 CD  20      Exit INT 20
0116 30          Reserve 30H bytes
```

3. Yes, it probably displayed garbage. End the sentence with a $ so the computer doesn't continue to try to display garbage until it finds a random ASCII 24_H ($). The screen echo should appear similar to that shown below.

```
-g Electronics Students!$
  Electronics Students!
  Program terminated normally -
```

Since one byte is eight bits, the maximum number is FF_H or 512 in decimal.

4. The memory dump for the example message is as follows. This information is used to build Table 4.

```
-d116
:0110                    30 16-45 6C 65 63 74 72 6F 6E          0.Electron
:0120  69 63 73 20 53 74 75 64-65 6E 74 73 21 24 0D 28   ics Students!$.(
```

Table 4

Addr.	ASCII	Character	
0116	30	Number of reserved characters	
0117	16	Number of characters typed	
0118	45	E	
0119	6C	l	
011A	65	e	
011B	63	c	
011C	74	t	
011D	72	r	
011E	6F	o	
011F	6E	n	
0120	69	i	
0121	63	c	
0122	73	s	
0123	20	Space	
0124	53	S	
0125	74	t	
0126	75	u	
0127	64	d	
0128	65	e	
0129	6E	n	
012A	74	t	
012B	73	s	
012C	21	!	
012D	24	$	End of message
012E	0D		End of characters entered

Procedure 3

3. 0200 DB 'hello'0D,0A

4. The line starting at address 0200 (above) must be added to the listing of Step 4.

5. Table 5

Addr.	ASCII	Character
0200	68	h
0201	65	e
0202	6C	l
0203	6C	l
0204	6F	o
0205	0D	Carriage return
0206	0A	Line feed

6.

Table 6

Address	Op code	Operand		Your comments
0100 8A	160002	MOV	DL,[0200]	Get first character
0104 B4	05	MOV	AH,05	Send it to printer
0106 CD	21	INT	21	
0108 8A	160102	MOV	DL,[0201]	Get second character
010C B4	05	MOV	AH,05	Send it to printer
010E CD	21	INT	21	
0110 8A	160202	MOV	DL,[0202]	Get third character
0114 B4	05	MOV	AH,05	Send it to printer
0116 CD	21	INT	21	
0118 8A	160302	MOV	DL,[0203]	Get fourth character
011C B4	05	MOV	AH,05	Send it to printer
011E CD	21	INT	21	
0120 8A	160402	MOV	DL,[0204]	Get fifth character
0124 B4	05	MOV	AH,05	Send it to printer
0126 CD	21	INT	21	
0128 8A	160502	MOV	DL,[0205]	Get CR character
012C B4	05	MOV	AH,05	Send it to printer
012E CD	21	INT	21	
0130 8A	160602	MOV	DL,[0206]	Get LF character
0134 B4	05	MOV	AH,05	Send it to printer
0136 CD	21	INT	21	
0138 CD	20	INT	20	Exit

9. Register CX must decrement to 0 to fall out of the loop. Each pass through the loop will decrement CX. Register BX holds the starting address of the message string. BX "points" to the message. Register DL gets the contents of the next memory location by looking at the location inside the square brackets held by BX. (Refer to Lesson 3's section on Register Indirect Address Mode.) Operation is on the memory location held in BX, not on BX itself. BX is incremented to move to the address of the next character to print.

10. NOTE: Message is printed to a printer, not to the CRT screen. Students must have a printer connected to the printer port for this experiment to work.

Table 7

```
0100 B9  1800    MOV CX,0018 Set up a counter
0103 BB  1A01    MOV BX,011A Load message start address
0106 8A  17      MOV DL,[BX] Now operate on address held in BX
0108 B4  05      MOV AH,05 Function to output character to printer
010A CD  21      INT 21    one at a time
010C 43          INC BX  Point to next ASCII character
010D E2  F7      LOOP back to address 0106 if CX not zero
010F CD  20      INT 20 Fall out of loop and Exit

011A  45  E
011B  6C  l
011C  65  e
011D  63  c
011E  74  t
011F  72  r
0120  6F  o
0121  6E  n
0122  69  i
0123  63  c
0124  73  s
0125  20  SPACE
0126  54  T
0127  65  e
0128  63  c
0129  68  h
012A  6E  n
012B  6F  o
012C  6C  l
012D  6F  o
012E  67  g
```

```
012F  79  y
0130  0D  CR
0131  0A  LF
```

Memory dump, starting at 011A, is used for Table 7.

```
0110                                 45 6C 65 63 74 72              Electr
0120  6F 6E 69 63 73 20 54 65-63 68 6E 6F 6C 6F 67 79   onics Technology
0130  0D 0A
```

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT IV: MICROPROCESSOR INSTRUCTIONS

LESSON 4: MORE MATH OPERATIONS

NOTE: Upon completion of this lesson, the student should be able to write stand-alone assembly language files and save them to disk.

I. Math operations

A. Addition

1. Math operations must be done often in any type of computer programming, whether for a sophisticated accounting program or for a robot or industrial process control.

2. The listing for a simple addition program is shown below.

```
MOV     AX,0000         ;Clear AX register
MOV     BX,0000         ;Clear BX register
MOV     AL,[0150]       ;Get 1st number
MOV     BL,[0151]       ;Get 2nd number
ADD     AL,BL           ;Add them together
MOV     [0152],AL       ;Store sum at 0152
INT     20              ;Exit
```

3. First the program will clear any "garbage" from registers AX and BX.

4. The two numbers to be added are stored at memory address 150 and 151. One number is moved into AL and the other number into BL.

5. Add the two numbers and the sum is left in AL.

6. The sum is then moved from AL to memory address 152.

7. Moving the sum is not necessary in a short demonstration program, but if the sum were to be used later in a long machine control program, for example, it should be stored in memory for later access.

8. The two numbers to be added, A and 8, are entered into memory locations 150 and 151.

9. Address 152 is cleared, waiting for the sum.

10. By placing the numbers in memory instead of using the register immediate mode, it is easy to change the values without writing the program again.

B. Rules for addition

1. If an addition produces a sum greater than FF_H (byte operation) or $FFFF_H$ (word operation), the carry flag is set.

2. If no carry occurred, the flag is reset.

3. If an addition causes a zero in the destination register, ZF will be set. If not zero, ZF will be reset.

4. If the addition causes the MSB of the destination register to be set, the SF will be set. Otherwise, SF will be clear.

5. If the addition causes an even number of set bits in the destination register, the PF will be set. Otherwise, PF will be reset.

C. Multiplication

1. Multiplication is really a series of repeated additions.

2. To multiply 4 x 3, add 4 to 4 to 4.

3. This same method can be used in a microprocessor to perform multiplication.

Example:
```
0100 MOV CX,0000      ;Clear CX then
0103 MOV AX,0000      ;Clear AX too
0106 MOV CL,[150]     ;Load the count register with 3
010A ADD AL,[151]     ;Add 4 to whatever is in AL
010E LOOP 10A         ;Do it 3 times
0110 MOV [155],AL     ;Then move the answer out to address 155
0113 INT 20           ;Done

0150 3
0151 4
0155 0
```

4. Trace through the listing with pencil and paper to see how multiplication by repeated addition works.

5. All microprocessors from the 8080 (6800, 6502 and Z80) use repeated addition to multiply.

6. The 8088 was the first microprocessor with built-in multiply and divide op codes.

7. The multiply instruction takes the form MUL register/memory.

8. The location of the second number and the result after the operation, is the A register.

9. As an example, MUL BL multiplies the contents of BL with the contents of AL and leaves the result in AX.

```
0100 MOV AX,0000      ;Clear the
0103 MOV BX,0000      ;registers first
0106 MOV AL,3         ;Place 3 in AL
0108 MOV BL,4         ;Place 4 in BL
010A MUL BL           ;Multiply 3 x 4
010C INT 20           ;Done
```

10. Using that mode is useful for constants, but you can also get the data from memory (variables).

```
0100 B80000     MOV     AX,0000
0103 BB0000     MOV     BX,0000
0106 A05001     MOV     AL,[0150]
0109 F6265101   MUL     BYTE PTR [0151]
010D A35501     MOV     [0155],AX
0110 CD20       INT     20
```

11. Notice that BYTE PTR gives the microprocessor operand size information.

12. The answer was moved from AX, a word register, and not the byte register AL, because the result of multiplying two 8-bit values may be up to 16 bits.

13. An unsigned byte from some source times an unsigned byte in AL register leaves the product in the AX register, the most-significant byte is in AH and the least-significant byte is in AL.

14. In the next listing word registers are used.

15. Notice that operand size information, WORD PTR, is included.

```
0100 MOV  AX,0000              ;Clear registers, not necessary
0103 MOV  BX,0000              ;but a good idea
0106 MOV  AX,[0150]            ;Get 4444 from memory and multiply
0109 MUL  WORD PTR [0160]      ;by 2222 so get a word, not byte
010D INT  20                   ;done

(Enter 4444 at address 150)
(Enter 2222 at address 160)
```

16. When multiplying words times a word in AX, up to 32-bit products are possible.

    a. The most-significant word is placed in DX.

    b. The least-significant word is placed in AX.

17. A quick way to square a number is to load it into AL the give the instruction MUL AL.

D. Subtraction

1. There is no borrow flag in the 8086/88.

2. The carry flag acts as the borrow flag during subtraction operations.

3. Trace through the following listing with pencil and paper.

```
0100 MOV  BL,23
0102 MOV  CL,20
0104 SUB  BL,CL
0106 SUB  BL,CL
0108 INT  20
```

4. Notice with the first subtraction the answer (3) is left in BL.

5. The next subtraction will cause a borrow.

6. Check the answer from the second subtraction with pencil and paper.

7. Is the answer correct?

8. Now change the SUB instructions to SBB (subtract with borrow).

```
0100 B323       MOV  BL,23
0102 B120       MOV  CL,20
0104 18CB       SBB  BL,CL
0106 90         NOP
0107 18CB       SBB  BL,CL
0109 90         NOP
010A CD20       INT  20
```

9. Trace the new listing with pencil and paper.

10. Note that the borrow flag is used as part of the answer.

11. If set, the result of the first subtraction will be 2 in BL and 1 in the flag.

12. If clear, the result of the subtraction was 3 in BL.

13. Recall that when using the SUB instruction the first answer was 3 in either case.

E. Rules for subtraction

1. Both SUB and SBB set the borrow flag when a larger number is subtracted from a smaller number.

2. The flag is cleared if a smaller number is subtracted from a larger number.

3. SBB includes the borrow flag in the subtraction, while SUB does not.

F. Division

1. Division is done on other microprocessors by repeated subtraction.

Example:

```
0100 MOV AX,0000
0103 MOV BX,0000
0106 MOV AL,[120]
0109 INC BX
010A SUB AL,[121]
010E JNZ 109
0110 MOV [122],BL
0114 INT 20

0120 0A
0121 02
```

2. Follow the logic through on paper. Notice that a counter is set up to keep track of how many subtractions were done.

3. The subtraction is done just before the jump instruction to test for a zero.

4. As in multiplication, the 8088 does have a divide instruction.

5. Trace this example with pencil and paper.

```
0100 MOV AX,0000
0103 MOV BX,0000
0106 MOV DX,0000
0109 MOV AX,[120]
010C MOV BX,[130]
0110 DIV BX
0112 INT 20

0120 0A

0130 02
```

6. Notice that a number in AX is divided, leaving the quotient in AL and any remainder in AH for byte operations.

7. For word operations the 16-bit operand is in DX--AX.

8. After division, the quotient goes in AX and the remainder in DX.

COMPUTE MATH OPERATIONS

NOTE: The evaluation for the following procedures is summarized at the end of the job sheet.

Equipment:

IBM-PC compatible computer
MS-DOS or PC-DOS
DEBUG

Procedure 1: Addition

1. Enter the following program.

```
MOV     AX,0000     ;Clear AX register
MOV     BX,0000     ;Clear BX register
MOV     AL,[0150]   ;Get 1st number
MOV     BL,[0151]   ;Get 2nd number
ADD     AL,BL       ;Add them together
MOV     [0152],AL   ;Store sum at 0152
INT     20          ;Exit
```

2. Enter any three numbers to be added in memory locations 150, 151 and 152.

3. Trace the program, watching the registers at each step, then examine address 150 through 152 for the result. Remember not to trace through INT 20.

4. Change the numbers to be added and trace through several times. Use numbers that will have a sum less than FF. Observe the carry flag after each addition. Recall that DEBUG will show NC for "no carry" occurred.

5. Now find the sum of FE and 2. Work this out first with pencil and paper. Notice that a carry must occur the same as when adding decimal numbers.

6. Place FE and 2 into the two memory locations and trace the program. Does the sum in register AL or memory location 152 agree with your calculation? _____ Explain.

7. Change the add (ADD) instruction to add with carry (ADC) and trace the program. Is the carry saved by the carry flag? _____

8. What other flags are affected by the ADD and ADC instructions?_____

Procedure 2: Multiplication

1. To multiply by repeated addition, enter the following listing.

```
0100 MOV CX,0000     ;Clear CX then
0103 MOV AX,0000     ;Clear AX too
0106 MOV CL,[150]    ;Load the count register with 3
010A ADD AL,[151]    ;Add 4 to whatever is in AL
010E LOOP 10A        ;Do it 3 times
0110 MOV [155],AL    ;Then move the answer out to address 155
0113 INT 20          :Done
```

```
0150 3
0151 4
0155 0
```

2. Trace the program, following the loop action.

3. Try multiplying different combinations of numbers for practice.

4. The multiply instruction takes the form MUL register/memory. The location of the second number and the result after the operation is the A register. As an example, MUL BL multiplies the contents of BL with the contents of AL and leaves the result in AX. Try this:

```
0100 MOV AX,0000      ;Clear the
0103 MOV BX,0000      ;registers first
0106 MOV AL,3         ;Place 3 in AL
0108 MOV BL,4         ;Place 4 in BL
010A MUL BL           ;Multiply 3 x 4
010C INT 20           ;Done
```

5. Using that mode is useful for constants, but you can also get the data from memory (variables).

```
0100 B80000     MOV     AX,0000
0103 BB0000     MOV     BX,0000
0106 A05001     MOV     AL,[0150]
0109 F6265101   MUL     BYTE PTR [0151]
010D A35501     MOV     [0155],AX
0110 CD20       INT     20
```

6. In the next listing, word registers are used. Notice that operand size information, WORD PTR, is included.

```
0100 MOV  AX,0000                ;Clear registers--not necessary
0103 MOV  BX,0000                ;but a good idea
0106 MOV  AX,[0150]              ;Get 4444 from memory and multiply
0109 MUL  WORD PTR [0160]        ;by 2222 so get a word, not byte
010D INT  20                     ;done
```

(Enter 4444 at address 150)
(Enter 2222 at address 160)

7. Notice when you trace this program that the 32-bit product is, indeed, stored DX--AX as shown in the register dump below.

```
AX=1908    BX=0000    CX=0000    DX=091A  SP=CFDE  BP=0000  SI=0000  DI=0000
DS=2778    ES=2778    SS=2778    CS=2778  IP=010D   OV UP DI PL NZ NA PO CY
2778:010D CD20      INT         20
```

8. A quick way to square a number is to load it into AL, then give the instruction MUL AL. Try it!

Procedure 3: Subtraction

1. Enter and trace the following listing.

```
0100 MOV  BL,23
0102 MOV  CL,20
0104 SUB  BL,CL
0106 SUB  BL,CL
0108 INT  20
```

2. Notice with the first subtraction the answer (3) is left in BL. The next subtraction will cause a borrow. Check the answer from the second subtraction with pencil and paper. Is the answer correct? _____

3. Now reset IP to 0100_H and trace again with the borrow flag set. Notice that the result of the first subtraction is again 3.

4. Now change the SUB instructions to SBB.

```
0100 B323      MOV     BL,23
0102 B120      MOV     CL,20
0104 18CB      SBB     BL,CL
0106 90        NOP
0107 18CB      SBB     BL,CL
0109 90        NOP
010A CD20      INT     20
```

5. Trace the new program. Note that the borrow flag is used as part of the answer. If set, the result of the first subtraction will be 2 in BL and one in the flag. If clear, the result of the subtraction was 3 in BL.

6. Recall that when using the SUB instruction the first answer was 3 in either case.

Procedure 4: Division

1. Division is done on other microprocessors by repeated subtraction. For example:

```
0100 MOV AX,0000
0103 MOV BX,0000
0106 MOV AL,[120]
0109 INC BX
010A SUB AL,[121]
010E JNZ 109
0110 MOV [122],BL
0114 INT 20

0120 0A
0121 02
```

Follow the logic through on paper. Notice that a counter is set up to keep track of how many subtractions were done. The subtraction is done just before the jump instruction to test for a zero.

2. Try different values and trace through the repeated subtraction process.

3. As in multiplication, the 8088 does have a divide instruction. Try this:

```
0100 MOV AX,0000
0103 MOV BX,0000
0106 MOV DX,0000
0109 MOV AX,[120]
010C MOV BX,[130]
0110 DIV BX
0112 INT 20

0120 0A

0130 02
```

4. Notice that a number in AX is divided, leaving the quotient in AL and any remainder in AH for byte operations. For word operations the 16-bit operand is in DX--AX. After division the quotient goes in AX and the remainder to DX.

5. Try numbers that will give remainders and trace through the program several times.

6. Divide any number by 0. What happened?

7. Turn off computer power and store diskettes.

Procedure:	Evaluation 3	2	1	N
1. Addition	—	—	—	—
2. Multiplication	—	—	—	—
3. Subtraction	—	—	—	—
4. Division	—	—	—	—
Average of above evaluations*	—	—	—	—

*This average is the student's partial evaluation for competencies I4-6.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT IV: MICROPROCESSOR INSTRUCTIONS

LESSON 5: ADVANCED PROGRAMMING

NOTE: Upon completion of this lesson, the student should be able to use shift, rotate and test instructions, as well as mask, set and toggle individual bits with Boolean instructions.

I. Bit shifts and rotations

A. Shift and rotate

1. Shift and rotate instructions differ from addition and subtraction in that 8- and 16-bit values are treated as collections of binary bits instead of numbers.

2. Consider the value in a register as a collection of 1s and 0s.

3. The position of the bits will be moved by the shift and rotate instructions.

B. Shift left instructions

1. The bits in an 8086/88 register, or in memory, can be shifted one or more bit positions left or right.

2. The instruction SHL shifts the bits in memory one bit position to the left.

a. A zero is shifted into bit position 0.

b. The contents of bit position 15 goes into carry.

c. The bit that was in carry is now lost.

```
                (Before SHL)
Carry           Register
x ← 1 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1  ← 0

                (After SHL)
Carry           Register
1     0 0 1 1 0 0 1 0 0 1 1 0 0 1 1 0
```

3. The format for a shift left is SHL r/m,1.

4. The 1 indicates shift the contents of the register or memory 1 bit left.

5. The 1 could also be CL.

a. Number of shifts determined by the contents of register CL

b. Shift from 1 to 256 bits

6. In shift left mnemonics, SHL means Shift Logical Left and SAL means Shift Arithmetic Left.

7. Both instructions do the same thing on a left shift.

a. Shift bits left equal to the count

b. Fill the bit positions on the right with zeros

8. What is the value of a shift left?

   a. Shift left 1 bit does a quick multiply by 2.

   b. Two consecutive left shifts will multiply by 4.

   c. Three shifts left will multiply by 8.

9. SHL is preferred over the MUL instruction for multiplying a number by a power of two since SHL will execute faster than MUL.

10. The multiplication instruction causes the microprocessor to perform a time-consuming series of shifts and adds.

11. Multiplication by something other than a multiple of two is also possible.

12. To multiply by AH (10 decimal), shift left to get a multiply by 2, store the result temporarily and shift left two more times. To get a multiply by 8 result, add the multiply by 2 result.

13. To multiply $3A_H$ by AH, trace the following example with paper and pencil.

```
MOV AX,3A ; Load 3A into register AX
SHL AX,1  ; Multiply by 2  AX = 2 X 3A
MOV BX,AX ; Store 2 X 3A temporarily
SHL AX,1  ; AX = 4 X 3A
SHL AX,1  ; AX = 8 X 3A
ADD AX,BX ; AX = (8 X 3A) + (2 X 3A) = A X 3A
```

C. Shift right instructions

1. Bits can also be shifted to the right.

2. SHR (shift logical right) will do the following.

   a. Shift the LSB out to the carry flag

   b. Shift zeros into the MSB position

   c. Fill all vacated bit positions on the left with zeros

3. SAR (shift arithmetic right) will shift the LSB out to the carry flag. The MSB and all vacated bits on the left will be filled with the original value of the MSB. The sign of the number will not be changed.

4. A division by two will result with each shift as long as a zero is shifted out to CF.

5. Bits shifted out of bit position 0 are the remainder.

D. Rotate

1. Rotate instructions are similar to shift instructions.

2. Rotate does not shift in a 0, but instead rotates in the bit that was rotated out the other side.

3. In the following example, AX is cleared and AL is loaded with 0F.

   a. The upper nibble has four zeros.

   b. The lower nibble has four set bits.

   c. Repeat ROR (rotate right) instruction four times.

   d. The nibble contents are exchanged.

4. Trace this program with pencil and paper.

5. Convert hex numbers to binary to see how it works.

```
MOV AX,0000
MOV AL,0F
ROR AL,1
ROR AL,1
ROR AL,1
ROR AL,1
```

6. ROL and ROR do not involve the carry.

7. The carry flag is used in the rotate sequence by RCL (rotate through carry left) and RCR (rotate through carry right).

8. The bit in carry is rotated into one end of the byte or word.

9. The bit at the other end is rotated into carry.

10. This is useful when multiple-of-two multiplication and division is needed on values larger than 16-bits.

11. For example, up to 32 bits held in BX (high word) and AX (low word) multiplied by two could be done in this way.

```
SHL  AX,1
RCL  BX
```

12. The SHL instruction shifts the contents of AX left, placing the MSB in carry.

13. The RCL instruction rotates the bit in carry into bit position 0 of BX.

14. All 32 bits were shifted left for a multiply by two.

II. Boolean instructions

A. Introduction

1. The microprocessor Boolean instructions below take the same operand form used by ADD and SUB.

   a. AND

   b. OR

   c. XOR

2. Logical functions are performed bit-by-bit, not byte-by-byte.

3. In other words, bit 3 of the source operand is compared to bit 3 of the destination operand, bit 7 is compared to bit 7, etc.

4. The action is similar to eight or 16 logic gates operating at once.

B. AND

1. An AND gate must have two logic 1s on the input to get a logic 1 output.

2. Any low at the input of an AND gate gives a low on the output.

3. One of the inputs, then, can act as a switch to either pass the other bit or inhibit the other bit from appearing on the gate output.

4. Consider the instruction AND AL,AH.

   a. Eight AND gates, one for each bit

   b. One input in AL

   c. One input in AH for each gate

5. The output of each gate will be in AL after the operation is completed.

   Example: If AL contained 37_H and it were ANDed with $0E_H$ in AH, the result in AL would be 06_H.

6. Look closely at the mask example below. The byte in AH has only three bits set: 1, 2 and 3.

7. The logic high bits act as switches turning on those bit positions.

8. This is a MASK that allows the condition of only those three-bit positions in AL to be present in the output byte.

9. All other bit positions are masked off by the zeros acting as switches, turning those bit positions off.

   Example of a mask:

   AL 0011 0111 (37_H) Byte in AL, bit 1 = 1, bit 2 = 1, bit 3 = 0
   AH 0000 1110 ($0E_H$) Mask off all but bits 1, 2, 3; force all other bits to 0
   AL 0000 0110 (06_H) Bit value of positions 1, 2, 3 retained; all others are forced to 0

10. When using SHR for power-of-two division, the remainder was the bits shifted out of bit position 0.

11. Modulo arithmetic deals with the remainder, not the quotient.

12. To find $C5_H$ modulo 4 (the remainder after dividing by 4), divide $C5_H$ by 4 by executing SHR twice.

13. This would shift two bits out of bit positions 0 and 1.

```
MOV AL,C5
SHR AL,1   Divide by 2
SHR AL,1   Divide by 2 again
```

14. A mask of 03_H would have only bit positions 0 and 1 set; all others are 0.

15. Find the remainder, C5 modulo 4, by adding the following instructions.

```
MOV AL,C5 1100 0101
AND AL,03  0000 0011  Mask off all but bits 0 and 1. Remainder is 01 left in AL
```

C. OR

1. An OR gate will have a logic high (1) on the output when an input is at logic high.

2. Any logic high on an input results in a logic high at the output.

   a. A logic high (1) ORd with a low (0) will be high.

   b. A logic high ORd with a high will be high.

3. The OR instruction can be used to set selected bits without disturbing the remaining bits.

4. Any bit ORd with a logic low (0) will not change.

   a. A low (0) ORd with a low will remain low.

   b. A low ORd with a high will remain high.

5. Consider the following.

   a. OR AL,AH

   b. If AL contained 09_H

   c. To set bits 1 and 5 but leave bits 0, 2, 3, 4, 6 and 7 unchanged, do the following.

```
                          MOV AL,09   0000 1001
                                        ↑    ↑
To set bits 1
and 5, use:               OR AH,22    0010 0010
The result in AL will be 2B.
The remaining bits (0, 2,
3, 4, 6 and 7) were not changed.      0010 1011
                                      ↑↑ ↑ ↑↑ ↑
```

   The remaining bits (0, 2, 3, 4, 6 and 7) were not changed.

D. XOR

1. Exclusive OR (XOR) means one or the other inputs (but not both) are high, giving a high output.

2. A low on the output always results when both inputs are high or both are low.

3. The XOR instruction is used to toggle selected bit positions.

   Example: Use 09_H again in this example

   XOR AL,AH

AL 0000	1001 09_H ↑↑↑↑	Complement (toggle) each bit of lower nibble
AH 0000	1111 $0F_H$	Use F (all 1s) in lower nibble
AL 0000	0110 06_H ↑↑↑↑	Result of XOR in AL Lower nibble toggled

4. XOR is a quick (two-byte) method of zeroing a register.

5. The instruction MOV AX,0000 will clear AX with three bytes.

6. The instruction XOR AX,AX will clear AX with two bytes.

E. TEST

1. Performs a logical AND

2. Will not change the bits

3. Updates flags

   Example:

   Move 0000 1101 ($0D_H$) into AL.
   Note that bit position 1 is a zero.
   The TEST instruction will test bit position 1.
   Because 02_H has only bit position 1, set (0000 0010).

4. TEST will AND bit position 1 without disturbing anything. Watch the zero flag as you trace through the program.

5. Then increment AL and test the same bit position. Again, observe the zero flag.

   Example of TEST instruction:

```
MOV     AL,0D                   ;Load AL with 0000 1101
MOV     [0150],AL               ;Move it to memory
TEST    BYTE PTR [0150],02      ;Test bit position 1
JZ      0120                    ;Jump if its a zero (see ZF)
NOP                             ;Fall to here if not zero
NOP                             ;You are here if it was a zero
INC     AL                      ;Now increment to 0000 1110
MOV     [0151],AL               ;Put it into memory
TEST    BYTE PTR [0151],02      ;Test the same bit position, if
JZ      0120                    ;bit position 1 was a zero, jump
NOP                             ;to 0120, but if it was not zero
INT     20                      ;go to 012D
```

6. A switch could be connected to one bit position at an address. Then use TEST to determine if the switch is on or off.

III. I/O ports

A. Introduction

1. There are two general methods of inputting and outputting data--I/O ports and memory-mapped I/O.

2. Some microprocessors, the MC6800 for example, only have memory-mapped I/O capability.

B. Memory-mapped I/O

1. With memory-mapped I/O, an input or output device can be placed at any memory address.

2. Data is written to the port or read from the port as any address would be, using any instruction that affects a memory location.

3. The advantage is that any number of devices at any memory location can be used and treated as just another memory location.

4. Any register that can access memory can access the memory-mapped port.

C. I/O ports

1. The advantage of I/O ports is that they have their own instructions.

a. IN and OUT

b. With port numbers

c. For much faster operation

2. The 8086/88 has 64K possible port locations, though usually only a few are actually used.

3. I/O ports are tied to the accumulator (AX) register for 16-bit data and to AL register for 8-bit data.

D. IN instruction

1. Copies a byte from a port number into AL using the form IN AL,PORT#

2. Copies a word from a port number into AX using the form IN AX,PORT#

3. Can also place port number in DX, then execute IN AL,DX

E. OUT instruction

1. The OUT instruction copies a byte from AL into a port number using the form OUT PORT#,AL.

2. It can also copy a word from AX into a port number using the form OUT PORT#,AX.

3. The port number can be placed in DX, then execute OUT DX,AL.

4. Note that only registers AX, AL and DX are used with IN and OUT instructions.

5. Flags are not affected.

DEMONSTRATE ADVANCED PROGRAMMING

NOTE: The evaluation for the following procedures is summarized at the end of the job sheet.

Equipment:

IBM-PC compatible computer
MS-DOS or PC-DOS
DEBUG

Procedure 1: Shift

1. Load and execute the following program. Observe register and flag results while tracing with DEBUG.

```
MOV  DX,F0CC
SAR  DX,1
SAR  DX,1
SHR  DX,1
SHR  DX,1
INT  20
```

Procedure 2: Rotate

1. Enter the following listing.

```
MOV AX,0000
MOV AL,0F
ROR AL,1
ROR AL,1
ROR AL,1
ROR AL,1
```

2. In DEBUG, convert the hex display to binary after each instruction to see how rotate works.

3. Try changing the instructions to rotate left and observe the operation.

```
SHL  AX,1
RCL  BX
```

4. The SHL instruction shifts the contents of AX left, placing the MSB in carry. The RCL instruction rotates the bit in carry into bit position 0 of BX. All 32 bits were shifted left for a multiply by two. Trace and observe the results.

Procedure 3: AND

1. Find $C5_H$ modulo 4 (the remainder after dividing by 4).

```
MOV AL,C5
SHR AL,1    Divide by 2
SHR AL,1    Divide by 2 again
```

2. Set a mask with the following instructions.

```
MOV AL,C5    1100 0101
AND AL,03    0000 0011        Mask off all but bits 0 and 1
                  ††          Remainder is 01 left in AL
```

3. Trace with DEBUG and observe the results.

Procedure 4: OR

1. Set selected bits without disturbing the remaining bits.

2. Enter:
```
MOV AL,09H
MOV AH,22H
OR AL,AH
```

3. Assemble and trace. Observe the results.

Procedure 5: XOR

1. Use XOR instruction to toggle selected bit positions.

2. Enter:
```
MOV AL,09H
MOV AH,0FH
XOR AL,AH
```

3. Assemble, trace and observe the results.

4. Assemble a short program to place a number in a register, then use XOR to clear it.

Procedure 6: TEST

1. The instruction TEST will perform a logical AND but will not change the bits while updating the flags. Enter the following listing.

```
MOV     AL,0D                ;Load AL with 0000 1101
MOV     [0150],AL            ;Move it to memory
TEST    BYTE PTR [0150],02   ;Test bit position 1
JZ      0120                 ;Jump if its a zero (see ZF)
NOP                          ;Fall to here if not zero
NOP                          ;You are here if it was a zero
INC     AL                   ;Now increment to 0000 1110
MOV     [0151].AL            ;Put it into memory
TEST    BYTE PTR [0151],02   ;Test the same bit position, if
JZ      0120                 ;bit position 1 was a zero, jump
NOP                          ;to 0120, but if it was not zero
INT     20                   ;go to 012D
```

2. Assemble, trace and observe the results.

Procedure 7: I/O

1. The loudspeaker in an IBM-PC or compatible is at port number 61_H. Let's make the speaker produce an audio tone. Bit 0 at port number 61_H is connected to an oscillator that needs to be turned off for now, so make sure bit zero is cleared to 0.

2. Bit 1 at port number 61_H is connected to a gate, which is connected to the loudspeaker. If bit 1 is turned on and off, the speaker sounds. The faster bit 1 is turned on and off, the higher the pitch will be.

3. The remaining six bits have nothing to do with the speaker; therefore, do not change them.

4. In summary, generating an audio tone involves the following steps.

   a. Find out how the bits are set at port 61_H.
   b. Turn bit 0 off.
   c. Turn bit 1 on and off by writing a word to the port.
   d. Use a time delay while switching bit 1 on and off.
   e. Do the whole procedure without changing bits 2 through 7.

5. First, find out how the bits at port 61_H are initially set. The IN instruction will copy the byte from port 61_H into register AL.

   0100 IN AL,61

6. Now mask off the two least significant bits while keeping the other six bits. Use the AND instruction; the xx represents the mask to be figured out. Mask byte should be _____ binary or ___ hex.

   0102 AND AL,xx

7. Bits 0 and 1 have been masked off. Bit 1 needs to be toggled on and off without changing any of the other bits. The XOR instruction will do this by using XOR with _____ binary or ___ hex. At this point it doesn't matter if bit 1 is a binary high or a low--just toggle it.

   0104 XOR AL,xx

8. Now send the processed byte in AL back out to port 61 with the out instruction.

   0106 OUT 61,AL

9. Move a count into the count register.

   0108 MOV CX,0190

10. Set up a time delay with the LOOP instruction, looping until the count register reaches zero.

    010_B LOOP 010_B

11. Do an unconditional jump back to the XOR instruction, where bit 1 will be toggled again.

    010D JMP 0104

12. The equal time delays (for speaker on and off) generate a square wave to the loudspeaker. The tone or pitch can be altered by changing the count in CX.

13. Put it all together, assemble it, save it to disk, trace it and then run the program.

14. After running the program, what needs to be added to it? _____

15. The full listing of the bytes needed in the above section is given below.

Tone generator listing

```
0100 IN      AL,61
0102 AND     AL,FC
0104 XOR     AL,02
0106 OUT     61,AL
0108 MOV     CX,0190
010B LOOP    010B
010D JMP     0104
```

16. Turn off computer power and store diskettes.

Procedure:	Evaluation 3	2	1	N
1. Shift	—	—	—	—
2. Rotate	—	—	—	—
3. AND	—	—	—	—
4. OR	—	—	—	—
5. XOR	—	—	—	—
6. TEST	—	—	—	—
7. I/O	—	—	—	—
Average of above evaluations*	—	—	—	—

*This average is the student's partial evaluation for competencies I4-6.

ANALYZE SIMPLE PROGRAMS

NOTE: For this exercise, the student will write a program to sound like a British siren.

Equipment:

IBM-PC compatible computer
MS-DOS or PC-DOS
DEBUG

Procedure:	Evaluation 3	2	1	N
1. Print your name on the screen.	—	—	—	—
2. Generate a two-tone (hi-low-hi-low) audio tone in speaker.	—	—	—	—
3. Each tone should last no more than two seconds before changing pitch to the next tone. Each high-low tone should repeat 10 times before the program ends.	—	—	—	—
4. Write and debug the siren program.	—	—	—	—
5. Assemble the program and save as a .COM file.	—	—	—	—
6. Show the program listing to the instructor. Run the siren program for the instructor.	—	—	—	—
7. Turn off computer power and store diskettes.	—	—	—	—
Average of above evaluations*	—	—	—	—

*This average is the student's final evaluation for competencies I4-6.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT V: COMPUTER SYSTEMS AND ADDRESS DECODERS

UNIT OBJECTIVE

Upon completion of this unit, the student should be able to construct and program simple address decoder circuits. This knowledge will be evidenced by completion of job sheets and the unit test to specified score.

SPECIFIC OBJECTIVES

After completion of the lessons in this unit, the student should be able to:

1.1 Define terms and definitions.

1.2 Describe two major differences between the 8086 and the 8088.

1.3 Recognize and describe the pin definitions of a maximum mode 8088 microprocessor.

1.4 Recognize and describe the pin definitions of a PC system bus.

1.5 Measure DC supply voltages with VOM. (competency I9 and H7-8, JS 1.1)

1.6 Measure bus activity with oscilloscope. (competency I9 and H7-8, JS 1.2)

2.1 Define terms and definitions.

2.2 Install interface breadboard. (partial competency I7, JS 2.1)

2.3 Explain the operation of an address decoder.

2.4 Construct and test interface decoder. (competency I7, JS 2.1)

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT V: COMPUTER SYSTEMS AND ADDRESS DECODERS

CONTENTS OF THIS UNIT

A. Objective sheet

B. References

C. Notes to the instructor

D. Lesson plans

1. Lesson 1: 8088 MICROCOMPUTERS

   a. Information outline

   b. Handout

      HO 1.1: 8086 and 8088 pinouts

   c. Assignment sheet

      AS 1.1: 8086 vs. 8088

   d. Answers to assignment sheet

   e. Job sheets

      JS 1.1: Measure supply voltages with VOM
      JS 1.2: Measure bus activity with oscilloscope

2. Lesson 2: I/O ADDRESS DECODE

   a. Information outline

   b. Transparency masters

      TM 2.1: Address decoder
      TM 2.2: I/O decoder (address 200_H)
      TM 2.3: I/O decoder (address 340_H)

   c. Handout

      HO 2.1: Decoder schematic

   d. Job sheet

      JS 2.1: Construct interface decoder

E. Unit test

F. Answers to unit test

REFERENCES

A. Books

1. Brey, Barry B. 8086/8088 Microprocessor: Architecture, Programming, and Interfacing. Columbus, OH: Merrill, 1987.

2. Brey, Barry B. The Z80 Microprocessor: Hardware, Software Programming and Interfacing. Englewood Cliffs, NJ: Prentice-Hall, 1988.

3. Coffron, James W. Microprocessor Programming, Troubleshooting, and Interfacing: The Z80, 8080, and 8085. Englewood Cliffs, NJ: Prentice Hall, 1988.

4. Dailey, Denton J. Small Computer Theory and Applications. New York: McGraw-Hill, 1988.

5. Gaonker, Ramesh. The Z80 Microprocessor: Architecture, Interfacing, Programming, and Design. Columbus, OH: Merrill, 1988.

6. Gillmore, Charles M. Microprocessors: Principles and Applications. New York: McGraw-Hill, 1989.

7. Hall, Douglas V. Microprocessors and Interfacing: Programming and Hardware. New York: McGraw-Hill, 1986.

8. Harman, Thomas L. The Motorola MC68020 and 68030 Microprocessors: Assembly Language, Interfacing, and Design. Englewood Cliffs, NJ: Prentice-Hall, 1989.

9. Hughes, Fredrick W. Microprocessor Technology: Theory and Experimentation. Englewood Cliffs, NJ: Prentice-Hall, 1989.

10. Kersey, Roger M. Personal Computer Operation and Troubleshooting. Englewood Cliffs, NJ: Prentice Hall, 1989.

11. Lafore, Robert. Assembly Language Primer for the IBM PC & XT. New York: The Waite Group, 1984.

12. Miller, Michael A. The 68000 Microprocessor: Architecture, Programming, and Applications. Columbus, OH: Merrill, 1988.

13. Norton, Peter. Inside the IBM PC. New York: Brady, 1986.

14. O'Connor, Patrick Joseph. Digital and Microprocessor Technology. 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1989.

15. Pasahow, Edward J. Microprocessor Technology and Microcomputers. New York: McGraw-Hill, 1988.

16. Tocci, Ronald J., and Lester P. Laskowski. Microprocessors and Microcomputers: Hardware and Software. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1987.

B. Curriculum guide

1. Brown, A. O. Advanced Microcomputer Service Technician. Stillwater, OK: Mid-America Vocational Curriculum Consortium, 1985.

NOTES TO THE INSTRUCTOR

A. Suggested approaches to teaching this unit

1. Demonstrate VOM and oscilloscope measurements.
2. Explain decoder circuits.
3. Construct and display interface decoder circuit.

B. Suggested student activities

1. Research additional interface circuits.
2. Trace decoder circuits on schematics of shop computers.

C. Supplemental materials

1. Please . . . don't give me any static (1/2" video). Presentation by Dan C. Anderson at electrostatic discharge (ESD) workshop for McDonnell Douglas Corp., 1985. (Available for free loan from Missouri Vocational Resource Center, 8 London Hall, University of Missouri-Columbia, 65211. Telephone 800/392-7217 or 314/882-9610.

2. "Electrostatic Discharge." Special supplement to Douglas Service 42 (Oct. 1985). Available from:

   McDonnell Douglas Corporation
   Editor, Douglas Service magazine (Lou Shaver)
   Douglas Aircraft Company (73-30)
   3855 Lakewood Boulevard
   Long Beach, CA 90846
   213/496-5001

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT V: COMPUTER SYSTEMS AND ADDRESS DECODERS

LESSON 1: 8088 MICROCOMPUTERS

NOTE: This lesson will address the purpose of the most-used pins on a microprocessor.

I. Terms and definitions

A. Queue--lining up in a row; like standing in line, waiting your turn

B. Breakout box--box or PC card used to make signal lines accessible for testing; some have built-in LED digital status indicators

II. The 8088 CPU

A. 8088 and 8086

1. The IBM-PC and most compatibles use the 8088 microprocessor.

2. Some PC clones and the lower-cost IBM PS/2 computers use the 8086 microprocessor.

3. Internally, both microprocessors have the same registers and the same op codes.

NOTE: Except for queue and data bus differences, this lesson will discuss the 8088 CPU.

4. There are two major differences--queueing and data buses.

a. The 8086 has a six-byte instruction queue, where the 8088 has a four-byte instruction queue.

b. The 8086 has a 16-bit data bus.

(1) 16-bit memory interface

(2) Switches between two banks of memory

(3) Must use signal NOT BHE to switch

c. 8088 has an eight-bit data bus.

(1) Eight-bit memory interface

(2) Bank switching is not needed

(3) Doesn't use NOT BHE signal

5. The difference in data bus width could be interpreted that the 8086 is twice as fast as the 8088. In general, this is <u>not</u> true.

6. Both microprocessors execute instructions from an internal queue that is filled as soon as there is an empty location.

7. It is possible for execution bus cycles to overlap instruction-fetch bus cycles.

8. If data is in 16-bit words, the 8088 takes two memory cycles, while the 8086 takes only one cycle.

9. If an application is heavily byte oriented, the 8088 will perform almost as fast as the 8086.

10. If an application is heavily word oriented, the 8088's speed is somewhat reduced (but not by 50%).

11. A definite advantage of the 8088 is that eight-bit support chips are readily available and lower in cost. Developed for the eight-bit 8085, the 8088 can be incorporated in product design just as IBM did when designing the PC.

B. Hardware model

1. To this point, the study of microprocessors has been primarily concerned with a programmer's model of the 8088.

   a. Internal registers

   b. Instruction codes

2. Next, the 8088 hardware model will be examined in terms of pin connections, signals and timing.

C. Minimum/maximum mode (HO 1.1)

1. Note on HO 1.1 that the 8088 (and the 8086) have two operating modes selected by pin 33 (MN, $\overline{MX}$).

   a. Minimum mode (MN)

   b. Maximum mode ($\overline{MX}$)

2. Minimum mode (MN) is on when pin 33 is strapped high.

   a. Pins are compatible with the Intel 8085 and the 8085 family of support chips.

   b. Bus commands from the 8088 are decoded and available.

   c. The request/grant interface is not supported.

   d. Math coprocessors cannot be used.

3. Maximum mode ($\overline{MX}$ is active when pin 33 is strapped low.

   a. A bus controller support chip is needed.

   b. The request/grant interface is supported.

   c. It is possible to attach coprocessors such as Intel 8087 math coprocessor.

4. In the IBM-PC and compatible design, pin 33 is strapped low for maximum mode.

5. Most of our discussion will be maximum mode pin definitions.

III. Microprocessor pin definitions

A. 8088 pin definitions

1. The 8088 microprocessor is packaged in a 40-pin DIP integrated circuit.

2. Each of the pins is defined for a specific function.

3. The name of each pin is given along with a short description of the use or function.

4. Pin names indicate active high except where a bar over the letter indicates that the pin is active low.

B. AD0-AD7 pin definition

1. Pins 9-16 are multiplexed at the beginning of the bus cycle (address bits A0-A7) and later in the bus cycle (data bits D0-D7).

2. In the PC design, CPU pins A0-A7 are latched, then re-powered to become A0-A7 on the system board bus.

3. In the same manner, pins D0-D7 are re-powered and assigned D0-D7 on the system board bus.

4. Even though at the microprocessor these eight pins are multiplexed, on the system bus they occupy separate pin locations.

C. A8-A15 pin definition

1. Pin 39 and pins 2-8 are memory and I/O address bits A8-A15.

2. These lines are not multiplexed and maintain stable output throughout the bus cycle.

3. In the PC design, these lines are latched and re-powered to become system bus address bits A8-A15.

D. A16/S3-A19/S6 pin definition

1. Pins 38 through 35 provide address bits A16-A19 at the beginning of the memory bus cycle.

2. Internal status of the 8088 is presented on these pins during the remainder of the cycle.

3. When S6 (pin 35) is held low, S5 (pin 36) gives the status of the interrupt enable flag. S4 (pin 37) and S3 (pin 38) are encoded to tell which segment register is being used for the bus cycle.

S4	S3	Segment
0	0	Alternate data segment
0	1	Stack segment
1	0	Code segment
1	1	Data segment

4. The PC design does not use status decoding. Address information is latched and re-powered to become system bus address bits A16-A19.

E. LOCK pin definition

1. Pin 29 is activated by a lock instruction and remains active until the end of the next instruction.

2. It is used in multi-master bus designs to tell other bus masters that they should not attempt to gain bus control.

3. The PC design is not a multi-master bus design and does not use this pin.

F. CLK pin definition

1. Pin 19 is the clock input and provides timing information for the 8088 microprocessor.

2. In the PC design, the clock signal comes from the 8284A clock chip (4.77MHz with a one-third duty cycle).

3. In "turbo" clones, this signal may be 8 or 10 MHz.

G. $\overline{RQ}$, $\overline{GT0}$ pin definition

1. Pin 31 is tied to the corresponding line of the 8087 math coprocessor socket in the PC design.

2. This bidirectional signal is used by other local bus masters to request the use of the bus.

3. The 8087 may request and take over the system bus to perform its operations.

H. $\overline{RQ}$, $\overline{GT0}$ pin definition

1. Pin 30 performs the same function as pin 31 but is on a lower priority level.

2. In the PC design, pin 30 is not used.

I. NMI pin definition

1. Pin 17 is an input line used to generate a non-maskable interrupt to the 8088 microprocessor.

2. A non-maskable interrupt cannot be ignored by the microprocessor. The CPU must stop what it is doing, store important information on the stack and service the interrupt request.

3. In the PC design, the NMI request is used to report parity errors in the system board memory and to halt the microprocessor. The interrupt request is masked outside the processor with a programmable port bit.

4. Interrupt requests from the math coprocessor and devices on the system bus are also sent to pin 17.

J. INTR pin definition

1. Input signal pin 18 is the maskable interrupt to the 8088 microprocessor.

2. In the PC design, pin 18 is connected to the 8259A interrupt controller. The 8259A expands this one input to eight system interrupt inputs.

K. READY pin definition

1. Input line, pin 22, is used to insert "wait" states in the 8088 microprocessor bus cycles, extending the cycle length.

2. In effect, the 8088 is slowed down while accessing I/O or memory that may be too slow for a normal bus cycle.

3. In the PC design, pin 22 is connected to the 8284A clock chip which synchronizes the 8088 with the system clock.

4. READY has several PC uses.

   a. Insert one wait state in all I/O port accesses

   b. Insert one wait state in all DMA cycles

   c. Provide a wait state function on the system bus

L. RESET pin definition

1. The reset input signal on pin 21 is used to halt the microprocessor.

2. In the PC design, the system power supply generates a "power good" signal when power supply voltages are at proper levels.

   a. The power good signal is passed to the 8284A clock chip.

   b. The clock chip passes the signal to pin 21, removing the reset and allowing the microprocessor to function.

M. QS0 and QS1 pin definitions

1. Pins 24 and 25 are output lines that give the status of the 8088's internal instruction queue.

2. In the PC design, these pins are connected to the 8087 math coprocessor socket so the 8087 can keep track of the status of the 8088's instruction queue.

N. $\overline{\text{TEST}}$ pin definition--Input pin 23 tested by the "wait for test" instruction

1. If $\overline{\text{TEST}}$ is low, execution continues.

2. If $\overline{\text{TEST}}$ is high, the 8088 waits.

O. $\overline{\text{S0}}$, $\overline{\text{S1}}$, $\overline{\text{S2}}$ pin definitions

1. Output pins 26, 27 and 28 present status information about the type of bus cycle that is to be performed.

2. This status is valid at the beginning of each bus cycle.

3. In PC design, the three pins are wired to the 8288 bus controller chip.

   a. The three bits are decoded by the 8288.

b. Output signals become control lines (IOR, IOW, MEMR, MEMW and ALE).

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Bus cycle type
0	0	0	Interrupt acknowledge
0	0	1	I/O port read
0	1	0	I/O port write
0	1	1	Halt
1	0	0	Code access
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive

IV. System bus signals

A. Card slots

1. The system bus on the IBM-PC and compatibles is implemented with 62-pin card slots. The PC has five slots, while the XT has eight slots.

2. Except power and ground buses, all of the signals are at TTL logic levels.

3. The bus is a superset of the 8088 microprocessor bus that is demultiplexed and re-powered.

4. Signals are added to support the following.

   a. Direct-memory (DMA) access

   b. Interrupts

   c. Timing and control of memory read and write, as well as I/O read and write

   d. Wait state generation

   e. Memory refresh for dynamic memory

   f. Error detection

B. OSC (oscillator)

1. The OSC line is output-only with a frequency of 14.31818 MHz for a period of about 70 ns at a duty cycle of 50%.

2. All other timing signals are generated from this signal.

3. OSC is the highest frequency present on the bus.

4. Divide 14.311818 MHz by four to get the 3.58 MHz standard broadcast, color burst signal.

C. CLK (clock)

1. CLK is derived from the OSC signal by dividing 14.31818 MHz by three to get the clock frequency of 4.77 MHz.

2. The clock period is 210 ns (high duration of 70 ns and low duration of 140 ns).

3. A typical bus-cycle is approximately 840 ns or four clock periods.

D. RESET DRV (reset driver)

1. RESET DRV is an output signal that is held active high during system power-up.

2. It remains active until all voltage levels have stabilized at their operating value; then, the line is brought low.

3. If a power supply voltage goes out of tolerance, the line is again brought active high to halt the system.

4. The signal is set active and inactive on the falling edge of the OSC signal.

5. It is used to provide a power-on reset to I/O devices, or any interface logic on the bus, to initialize them to a known state at system start-up.

E. A0-A19

1. The 20 address bits are output-only and are used for addressing I/O and memory plugged into the system card slots.

2. During DMA they are released by the microprocessor and driven by the DMA logic.

F. D0-D7: eight bidirectional data lines that are used to send data to and from the 8088 microprocessor, memory, I/O and I/O ports

G. ALE (address latch enable)

1. ALE is an output-only signal from the 8288 bus controller.

   a. The ALE signal goes active high just before the address is valid.

   b. It falls to inactive low just after the address bus is valid.

2. The signal is used to latch the address information from the address/data bus of the 8088 microprocessor.

3. The system data bus does not contain address information.

4. ALE cannot be used to demultiplex addresses from the system data bus.

5. The ALE signal is a good oscilloscope sync point when observing 8088-initiated bus cycles since it starts at the beginning of each bus cycle.

6. During DMA, ALE is not active.

H. I/O CH CK (I/O channel check)

1. When set low, this input-only signal will report error conditions on the interface cards to the 8088 via the NMI.

2. Parity checks on memory expansion cards are reported on this line.

I. I/O CH RDY (I/O channel ready)

1. An input-only signal, used to extend bus cycle length to enable slower memory or I/O ports, can be used with the system.

2. When a card wants to extend the bus cycle beyond the normal four clocks, it will force I/O CH RDY low when it receives a MEMR, MEMW, IOR or IOW command.

J. IRQ2-IRQ7 (interrupt requests 2-7)

1. These 6 input-only signals go directly to the 8259A interrupt controller on the system board.

2. The ROM BIOS programs initialize the 8259A so that IRQ2 is the highest priority and IRQ7 is the lowest priority.

3. If the priority level is not masked, a rising edge signal will generate an interrupt request to the 8088 microprocessor.

4. Once the request occurs, it must remain active high until the 8088 issues an INTA (interrupt acknowledge) signal.

K. IOR (I/O read)

1. IOR is an output from the 8288 bus controller to tell the I/O ports the present 8088 initiated bus cycle is an I/O read cycle. The address on the address bus is an I/O address.

2. This signal is active low.

L. IOW (I/O write)--An active low output signal from the 8288 bus controller tells the selected I/O port to accept data on the data bus.

M. MEMW (memory write)--An active-low output signal from the 8288 bus controller tells the memory which address is on the address bus and to accept data from the data bus placed there by the microprocessor during a write operation.

N. MEMR (memory read)--An active-low output signal from the 8288 bus controller that tells a memory location to place data on the data bus for the microprocessor to read

O. DRQ1-DRQ3 (direct-memory access request 1-3)--Active high lines when used by the interface to request a DMA operation between itself and memory without the 8088 intervention

P. DACK0-DACK3 (direct-memory access acknowledge 0-3)

1. Four active-low signals issued by the 8237-5 DMA controller

2. Indicates that the corresponding DRQ has been honored and the DMA controller will take the bus control away from the microprocessor and proceed with the data transfer

Q. AEN (address enable)

1. An output-only, active-high signal issued by the DMA controller to indicate a DMA bus cycle is in progress

2. AEN signal uses

a. Disable the 8088 microprocessor data, address and control buses from the system bus

b. Enable the address and control bus from the DMA controller

3. Used on the system bus to disable I/O address decodes to avoid conflicts between I/O and DMA

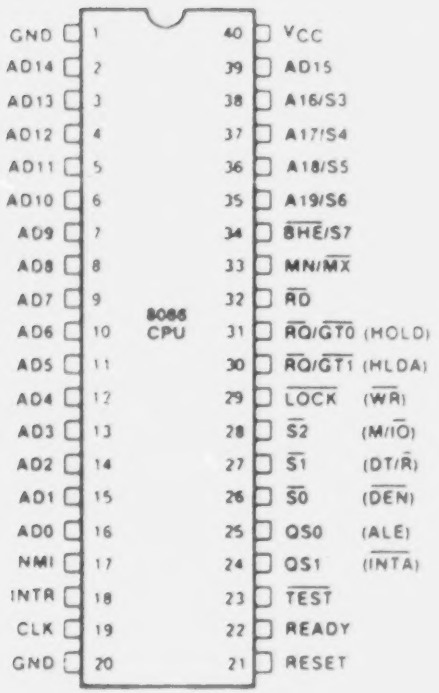
R. TC (terminal count)

1. An output-only, active-high signal from the DMA controller that indicates one of the DMA channels has reached its preprogrammed number of transfer cycles

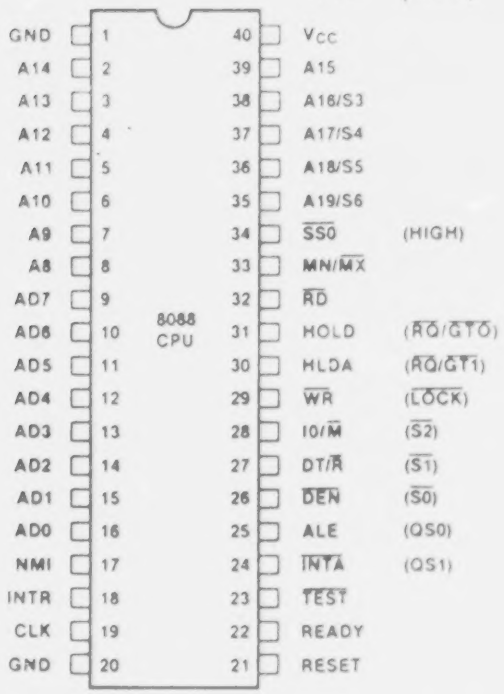
2. Used to terminate a DMA block data transfer

8086 AND 8088 PINOUTS

8086



8088



8086 VS. 8088

Answer the following questions about the PC bus and the 8086/88 microprocessor.

1. How many pins are available on the IBM PC card slot? _____
2. How many card slots are available on the PC? _____
3. How many card slots are available on the XT? _____
4. What is the significance of the OSC signal on the PC bus?
5. Describe the function of bus lines D0 through D7.
6. What signal would be a good oscilloscope sync point to observe the 8088 bus cycle?
7. In the PC design what is the power good signal used for?
8. What size and type of I.C. is the 8088 package?
9. In the IBM PC design is the 8088 operated in minimum or maximum mode? _____
10. Name one advantage of the 8088 over the 8086.
11. What are the differences in queue between the 8088 and the 8086?
12. What are the differences in internal registers between the 8088 and the 8086?
13. What are the differences in data bus and memory interface between the 8088 and the 8086?

ANSWERS TO AS 1.1

1. 62 pins
2. Five card slots are available.
3. Eight card slots are available.
4. All other timing signals are generated from this signal. OSC is the highest frequency present on the bus.
5. The eight data lines are bidirectional and are used to send data to and from the 8088 microprocessor, memory, I/O and I/O ports.
6. The ALE signal is a good oscilloscope sync point when observing 8088-initiated bus cycles since it starts at the beginning of each bus cycle.
7. The microprocessor is halted by the reset signal when the PC is first turned on. When the power supply voltages stabilize, a power good signal removes the reset signal, allowing the computer to start.
8. The 8088 microprocessor is packaged in a 40-pin DIP integrated circuit.
9. Maximum mode
10. A definite advantage of the 8088 is that it uses eight-bit support chips that are readily available and lower in cost.
11. The 8086 has a six-byte instruction queue. The 8088 has a four-byte instruction queue.
12. None; they both have 16-bit internal registers.
13. The 8086 has a 16-bit data bus and 16-bit memory interface. The 8088 has an eight-bit data bus and eight-bit memory interface.

MEASURE SUPPLY VOLTAGES WITH VOM

Equipment:

IBM PC or compatible
Service manual or "Computerfacts"
Digital VOM
PC bus card extender

		Evaluation 3	2	1	N
Procedure 1: Measure power supply voltages					
1.	Turn computer power off. Remove computer case.	—	—	—	—
2.	Locate power supply connector(s) on system board. Use schematics and make a chart of power supply DC voltages.	—	—	—	—
3.	Turn computer power on. Measure and record power supply voltages.	—	—	—	—
Procedure 2: Measure expansion slot voltages					
1.	Turn computer power off. Use schematics to locate and identify expansion bus ground pins and pins that carry DC power. Make a chart of pin numbers and voltages expected.	—	—	—	—
	CAUTION: Use card extenders to avoid probing on the system board and card slots. A slip of probe could short out the system board.				
2.	Insert card extender and turn on power.	—	—	—	—
	NOTE: A card extender is an example of a "breakout box" used to make signal lines accessible for test.				
3.	Measure and record DC voltages on the card extender.	—	—	—	—
Procedure 3: Measure RESET DRV (reset driver) line					
1.	Turn computer off. Locate RESET DRV line on the card extender. Connect DVOM to RESET DRV line.	—	—	—	—
2.	Turn computer power on and observe DC voltage present on line. The DC voltage should hold near 5V for a few seconds, then drop (signalling that power is normal) and reset any devices monitoring the line.	—	—	—	—
3.	Ask the instructor to check that your circuit is constructed correctly.	—	—	—	—
4.	Ask the instructor to check that the circuit was analyzed correctly.	—	—	—	—
5.	Discuss the project with the instructor in a knowledgeable manner.	—	—	—	—
6.	How long did it take to perform these steps? _____ hrs., _____ min.	—	—	—	—
7.	Turn equipment off; store equipment and tools properly.	—	—	—	—

Average of above evaluations* _____

*This average is the student's partial evaluation for competencies I9 and H7-8.

MEASURE BUS ACTIVITY WITH OSCILLOSCOPE

Equipment:

IBM PC or compatible
Service manual or "Computerfacts" for computer
Oscilloscope, dual trace
Extender card for expansion bus

Procedure:		Evaluation 3	2	1	N
1.	Turn computer power off. Remove computer case.	—	—	—	—
2.	Install card extender.	—	—	—	—
	CAUTION: Use card extenders to avoid probing on the system board and card slots. A slip of probe could short out the system board.				
3.	Locate ALE, CLOCK and OSC lines on card extender.	—	—	—	—
4.	Set oscilloscope on External Sync. Connect a sync probe from the External Sync. jack to ALE on the extender card.	—	—	—	—
5.	Turn on power to computer and oscilloscope. Observe waveform at CLOCK line on channel A of the oscilloscope.	—	—	—	—
6.	Observe waveform at OSC line on channel B of the oscilloscope.	—	—	—	—
7.	Adjust oscilloscope for stable display.	—	—	—	—
8.	Sketch the observed waveforms. Label with test point observed, volts P-P and time base.	—	—	—	—
9.	Ask the instructor to check that your circuit is correctly constructed.	—	—	—	—
10.	Was the circuit correctly analyzed?	—	—	—	—
11.	Discuss the project with the instructor in a knowledgeable manner.	—	—	—	—
12.	How long did it take to perform these steps? (_____ hrs., _____ min.)	—	—	—	—
13.	Turn equipment off, store equipment and tools properly.	—	—	—	—
Average of above evaluations*		—	—	—	—

*This average is the student's partial evaluation for competencies I9 and H7-8.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT V: COMPUTER SYSTEMS AND ADDRESS DECODERS

LESSON 2: I/O ADDRESS DECODE

NOTE: Upon completion of this lesson, the student should be able to construct and test an I/O port decoder.

I. Terms and definitions

A. Kludge--to construct or breadboard a circuit; also refers to the type of board used to construct these circuits

B. Decoder--logic circuit used to decode or respond to only one of many address codes on the address bus

C. Buffer--digital circuit that buffers or separates the computer bus from I/O circuitry

II. Breadboards

A. Direct connect breadboard

1. The IBM PC has five expansion slots on the system board.

2. The IBM PC/XT and compatibles have eight expansion slots on the system board.

3. All data, address and most control bus lines are available on the expansion slots.

4. Several Kludge boards allow circuits to be constructed and connected directly to the system lines.

5. Breadboards are available for several types of circuit construction.

a. Wire wrap

b. Solderless

c. Soldered

6. If a breadboarded interface experiment were to short out, serious damage could result on the system board.

B. Buffered breadboard

1. To prevent damage yet allow interface experiments, several Kludge boards are available that buffer or isolate system lines from the experimental circuits.

2. The PB-88/4 Breadboarding Design Station is available from Global Specialties. Other similar units are available.

3. The Breadboarding Design Station consists of three parts.

a. A system bus buffer interface card

b. The breadboard console which contains a solderless breadboard area

c. Two cables to connect the two boards together

C. The Breadboarding Station

1. There are two reasons for buffering the system signals.

   a. The PC can only supply limited current, about two TTL fan outs, per expansion slot.

   b. Isolate the experimental circuit from the system lines so that a short in the experimental circuit will not affect the computer system.

2. There are three sets of signals that are needed for interface experiments.

   a. Data, D0-D7

   b. Address, A0-A19

   c. Control

      (1) MEMR & MEMW
      (2) IOW & IOR
      (3) RESET
      (4) CLOCK & OSC

D. Unique addresses

1. Since address and data signals are present on the breadboard at all times, any circuit that is breadboarded would be present at each address examined by the microprocessor.

2. Each address must be a unique address, much like a street address and house number in a city.

3. Only one address must exist or be enabled at any given time.

4. A 64K memory chip, for example, contains 65,535 separate addresses.

5. A computer will usually have more than one memory chip.

6. The exact address is decoded by selecting which chip to enable, then selecting the proper address within the chip.

7. Only bits A0-A9 are used for I/O addressing in the IBM PC design, allowing I/O addresses 200_H-$3FF_H$ out on the system expansion slots.

Address range	Partly used by
200_H-$23F_H$	Game control adapter
240_H-$27F_H$	Second printer port adapter
280_H-$2BF_H$	Not used presently
$2C0_H$-$2FF_H$	Second printer port adapter
300_H-$33F_H$	Prototype card
340_H-$37F_H$	Printer port adapter
380_H-$3BF_H$	Monochrome adapter
$3C0_H$-$3FF_H$	Color graphics adapter
	5 1/4" disk drive controller
	Serial port adapter

8. Bits A10-A15 may be active on the bus but are ignored during direct I/O reads and writes.

9. Bits A16-A19 are disabled during direct I/O reads and writes.

E. I/O port addresses

1. Some of the I/O ports are already in use.

   a. Video graphics card

   b. Disk drive controller card

   c. Printer port adapter card

2. Any circuits that are breadboarded can use any I/O port address but cannot conflict with existing port addresses.

3. Addresses 300_H-$33F_H$ are specifically reserved for a prototype card or breadboard.

4. To avoid conflicts, place the breadboard within these addresses.

5. Four unique addresses are needed.

6. Design a decoder for hex addresses 300_H-303_H.

F. Address decoder **(TM 2.1)**

1. The SN74LS138 I.C. is a three-line to eight-line decoder.

2. The chip is enabled any time that G1 (pin 6) is at a logic high, and both G2B (pin 5) and G2A (pin 4) are asserted logic low.

3. The outputs are normally a logic high.

   a. Y0 through Y7

   b. Pins 15 through 9 and pin 7

4. When the 74LS138 is enabled, a three-bit code applied to A, B and C (pins 1, 2 and 3) will decode, pulling one of the output pins low.

5. Three bits have eight distinct binary combinations--0 through 7.

Address in			Output
C	**B**	**A**	**LOW**
0	0	0	Y0
0	0	1	Y1
0	1	0	Y2
0	1	1	Y3
1	0	0	Y4
1	0	1	Y5
1	1	0	Y6
1	1	1	Y7

6. In order to read or write to a breadboard I/O address, either the I/O read or I/O write line must be active low.

7. By placing both lines at NAND gate inputs, one line will be low and the other line will be high during an I/O operation.

8. A low and a high on the NAND input will place a logic high at G1.

9. At times other than I/O operations, both the I/O read and write lines will be high, placing a logic low on G1, disabling the decoder.

10. The AEN (address enable) line goes active high only during DMA transfers.

11. While the microprocessor is controlling the bus, AEN stays low.

12. Use AEN to pull G2A low to be sure that the breadboard is addressed only when the microprocessor executes a read or write.

13. The allowed direct I/O addresses start at 200_H (512 decimal).

14. The binary bit pattern for 200_H is below.

```
bit 9       bit 0
  |           |
0010 0000 0000 binary
  2    0    0   hex
```

15. Notice that bit 9 is high starting at 200_H and is low for any number less than 200_H.

16. Placing bit 9 (A09) at both inputs of a 7400 NAND gate will invert the high to a low and pull G2B low.

17. When bit 9 is a low the decoder will be disabled.

18. In summary, when address bit 9 is high, DMA does not have control. An I/O read or write is requested by the microprocessor. The decoder will decode address bits A6, A7 and A8 (see table).

G. Examples **(TM 2.2)**

1. In TM 2.2, address 200_H would pull output pin Y0 low.

2. This pin would be used as an active-low, chip-select signal to respond to a joy stick.

Address in			Output
C	**B**	**A**	**Low**
0	0	0	Y0 <<<
0	0	1	Y1
0	1	0	Y2
0	1	1	Y3
1	0	0	Y4
1	0	1	Y5
1	1	0	Y6
1	1	1	Y7

Decoder	Address range	Partly used by
Y0	200_H-$23F_H$	Game control adapter
Y1	240_H-$27F_H$	Second printer port adapter
Y2	280_H-$2BF_H$	Not presently used
Y3	$2C0_H$-$2FF_H$	Second printer port adapter

Y4	300_H-$33F_H$	Prototype card
Y5	340_H-$37F_H$	Printer port adapter
Y6	380_H-$3BF_H$	Monochrome adapter
Y7	$3C0_H$-$3FF_H$	Color graphics adapter
		5 1/4" disk drive controller
		Serial port adapter

3. The bit pattern for 340_H is as follows.

```
bit 9       bit 0
  |           |
0011 0100 0000 binary
  3    4    0   hex
```

4. Bits A6 and A8 will go high **(TM 2.3)** for bit pattern 101 decoded as 5 decimal.

5. Output pin Y5 would pull low (see above table) and place the printer adapter card on the data bus.

Address in			Output
C	**B**	**A**	**Low**
0	0	0	Y0
0	0	1	Y1
0	1	0	Y2
0	1	1	Y3
1	0	0	Y4
1	0	1	Y5 <<<
1	1	0	Y6
1	1	1	Y7

Decoder	Address range		Partly used by
Y0	200_H-$23F_H$		Game control adapter
Y1	240_H-$27F_H$		Second printer port adapter
Y2	280_H-$2BF_H$		Not used presently
Y3	$2C0_H$-$2FF_H$		Second printer port adapter
Y4	300_H-$33F_H$		Prototype card
Y5	340_H-$37F_H$	<<	Printer port adapter
Y6	380_H-$3BF_H$		Monochrome adapter
Y7	$3C0_H$-$3FF_H$		Color graphics adapter
			5 1/4" disk drive controller
			Serial port adapter

6. To address the prototype breadboard, the bit pattern for 300_H **(TM 2.1)** is as follows.

```
bit 9       bit 0
  |           |
0011 0000 0000 binary
  3    0    0   hex
```

7. Bit A8 will go high **(TM 2.1)**, bits A6 and A7 high for bit pattern 100 decoded as 5 decimal.

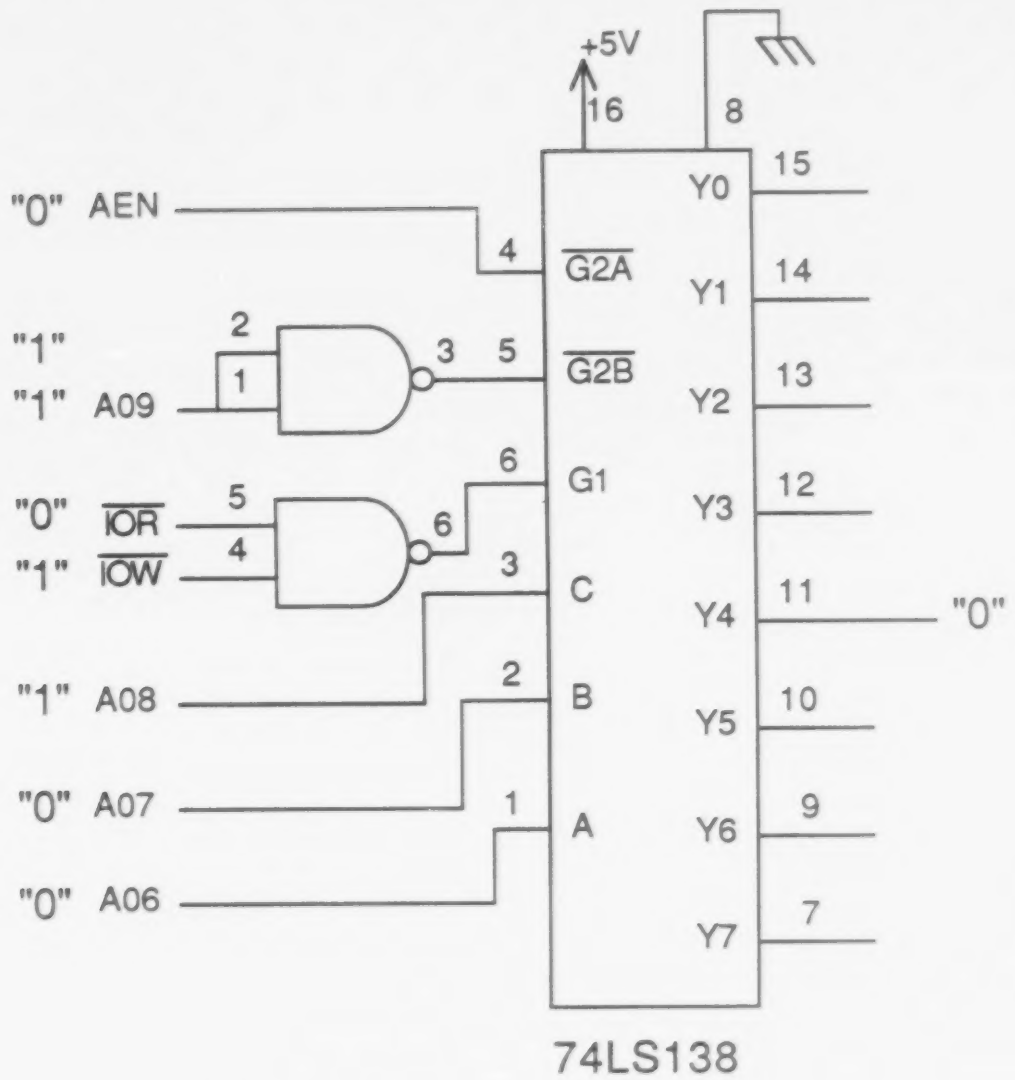
8. Output pin Y4 would pull low and place the prototype breadboard on the data bus.

Address in			Output
C	**B**	**A**	**Low**
0	0	0	Y0
0	0	1	Y1
0	1	0	Y2

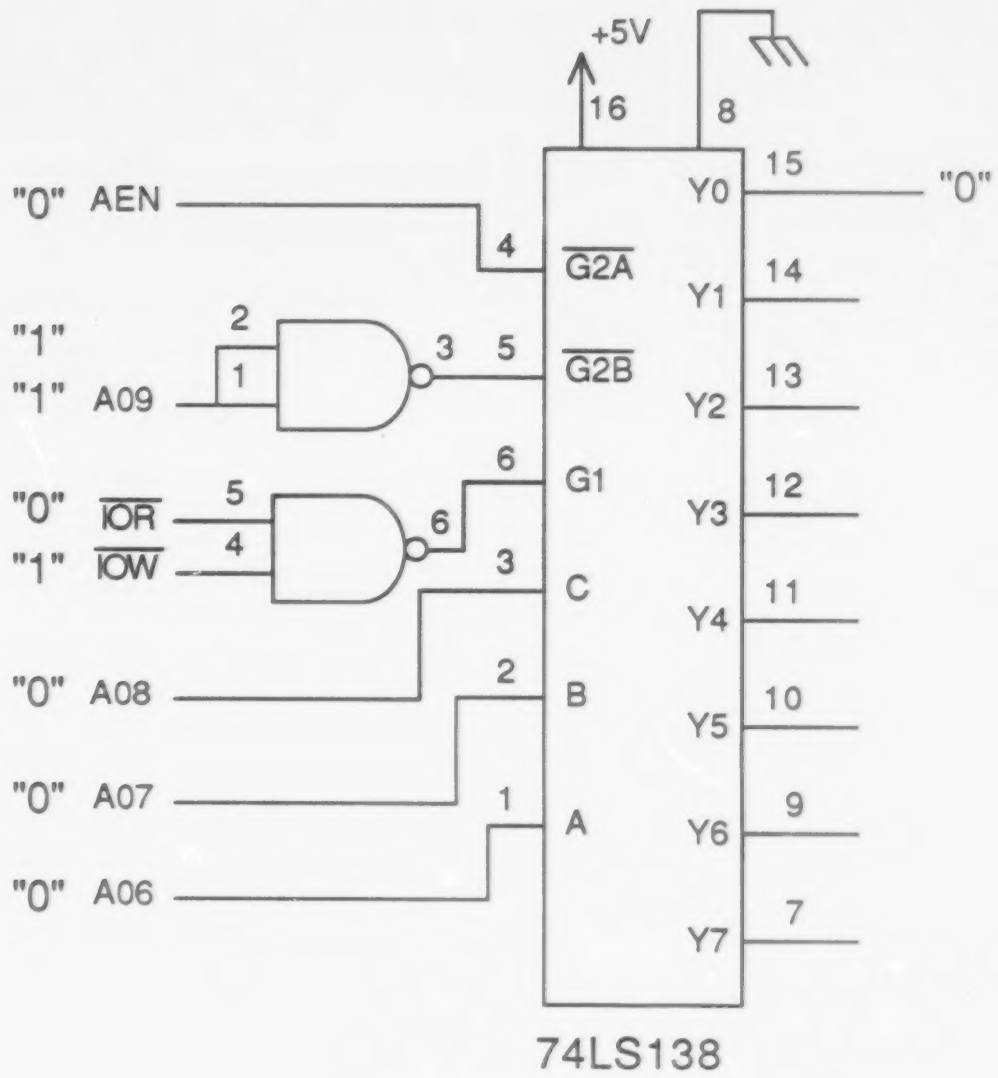
0	1	1	Y3	
1	0	0	Y4	<<<
1	0	1	Y5	
1	1	0	Y6	
1	1	1	Y7	

Decoder	Address range	Partly used by
Y0	200_H-$23F_H$	Game control adapter
Y1	240_H-$27F_H$	Second printer port adapter
Y2	280_H-$2BF_H$	Not used presently
Y3	$2C0_H$-$2FF_H$	Second printer port adapter
Y4	300_H-$33F_H$ <<	Prototype card
Y5	340_H-$37F_H$	Printer port adapter
Y6	380_H-$3BF_H$	Monochrome adapter
Y7	$3C0_H$-$3FF_H$	Color graphics adapter
		5 1/4" disk drive controller
		Serial port adapter

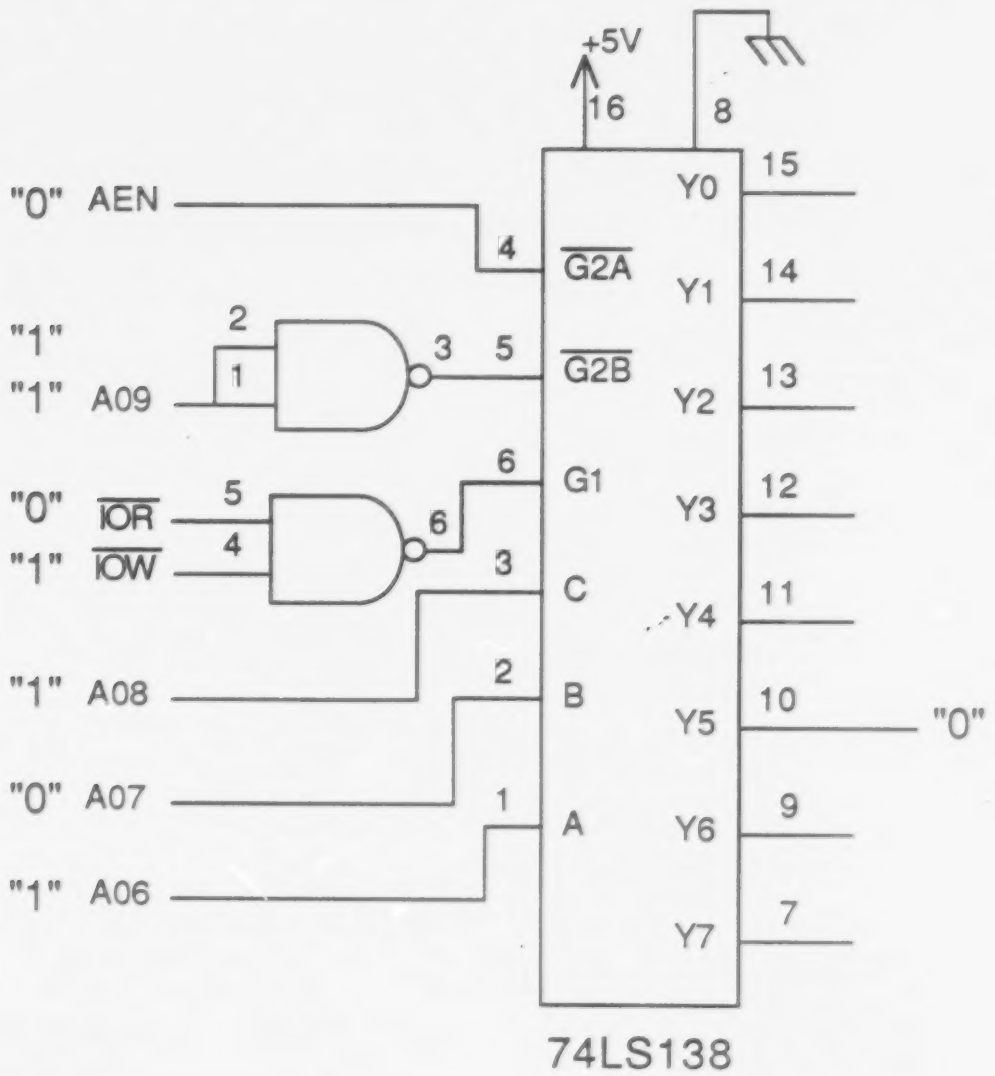
ADDRESS DECODER



I/O DECODER (ADDRESS 200_H)

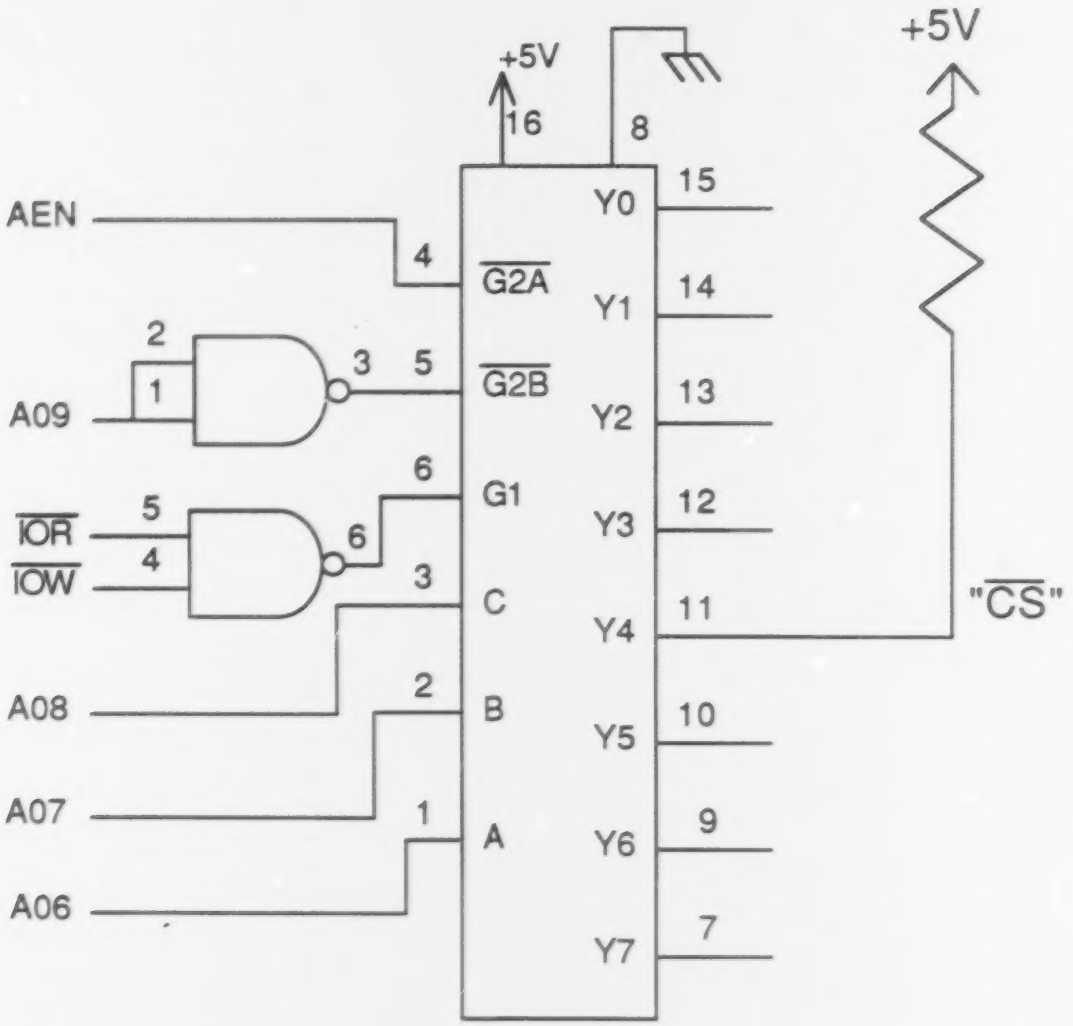


I/O DECODER (ADDRESS 340_H)



DECODER SCHEMATIC

ADDRESS 340



Chip select I/O decoder

230

CONSTRUCT INTERFACE DECODER

INSTRUCTOR'S NOTE: Remind students of the hazards and damage associated with electrostatic discharge (ESD). (See supplemental materials.)

Equipment:

HO 2.1: Decoder schematic diagram
IBM PC or compatible
DOS and GWBASIC diskette
I/O breadboard and schematic or manual
74LS138
74LS00
10K resistor
20-gauge solid hook-up wire, assorted colors
Digital VOM
Oscilloscope

		Evaluation 3	2	1	N
Procedure 1: Installation of I/O breadboard					
1.	Boot the computer to make sure that it works properly. Turn the computer off. Open the PC case.	—	—	—	—
	WARNING: Never make any hardware changes with computer power on.				
2.	Ground yourself on the PC power supply module to prevent electrostatic discharge (ESD).	—	—	—	—
3.	Install the I/O buffer card in the PC expansion bus.	—	—	—	—
4.	Turn the computer on.	—	—	—	—
	NOTE: If the computer does not boot, turn power off and notify the instructor.				
5.	Measure the DC voltages present on the I/O board. They are +5v, -5v, +12v, -12v. All voltages should be normal (within 10% of marked value).	—	—	—	—
6.	Examine both CLOCK and OSC signals for activity on the breadboard with an oscilloscope.	—	—	—	—
Procedure 2: Construct decoder (HO 2.1)					
1.	The two outer strips of each breadboard socket (if available) should be used for ground and +5V buses. Use red wire for the V_{cc} lines and black wire for the ground lines. Place +5V on the extreme outer row and ground on the row nearest the breadboard socket.	—	—	—	—
2.	Straddle the breadboard socket trench with the SN74LS138. Connect a red wire from pin 16 to the +5V bus. Connect a black wire from pin 8 to the ground bus.	—	—	—	—

3. Place an SN74LS00 just above the decoder chip. Connect a red wire from pin 14 to the +5V bus. Connect a black wire from pin 7 to the ground bus.

   Continue wiring the I.C. pins to the breadboard bus. Alternate colors of wires to make troubleshooting and circuit tracing easier. Keep wires neat and close to the board. Do not run wires over an I.C. — — — —

4. Turn the computer on. — — — —

   NOTE: If the computer does not boot, turn power off and check the wiring.

Procedure 3: Test decoder — — — —

1. Attach a logic probe to $\overline{CS}$ (chip select line) on the breadboard. The probe should show a binary 1 (high).

2. Run GWBASIC by typing GWBASIC at the DOS "A>" prompt. Enter "OUT &H300,1". This should cause the logic probe to blink to a binary 0 (low) once. If the probe did not show low, enter the BASIC command again. (If low is still not seen, contact the instructor.) The low indicates that I/O port 300_H has been enabled. — — — —

3. Does chip select pull low? (If it does, check-out is complete.) — — — —

4. Ask the instructor to check that the circuit was constructed correctly. — — — —

5. Was the circuit analyzed correctly? — — — —

6. Discuss the project with the instructor in a knowledgeable manner. — — — —

7. How long did it take to perform these steps? (_____ hrs., _____ min.) — — — —

8. Turn equipment off; store equipment and tools properly. — — — —

NOTE: Save the constructed I/O decoder for use with a later job sheet.

Average of above evaluations* — — — —

*This average is the student's partial evaluation for competency I7: Describe interfacing techniques.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT V: COMPUTER SYSTEMS AND ADDRESS DECODERS

UNIT TEST

Select the option which answers the question or completes the statement.

___ 1. The IBM XT and most clones have how many card slots on the motherboard?

a. 2
b. 3
c. 5
d. 8
e. 12

___ 2. Which bus signal synchronizes all PC bus activity?

a. ALE
b. DMA
c. OSC
d. IOW
e. IOR

___ 3. Which bus lines carry data to and from the 8088 microprocessor?

a. D0-D7
b. D1-D8
c. D0-D15
d. A0-A7
e. A0-A15

___ 4. How many pins are in an 8088 microprocessor's pin package?

a. 8
b. 22
c. 40
d. 62
e. 100

___ 5. When the power is turned on, the PC power supply stabilizes and generates a power good signal. What signal is removed, allowing the computer to start?

a. Reset
b. Control-alternate-delete
c. ALE
d. IOW
e. IOR

___ 6. What kind of external data bus and internal registers does an 8088 microprocessor have?

a. Eight-bit external, eight-bit internal
b. Eight-bit external, 16-bit internal
c. 16-bit external, eight-bit internal
d. 16-bit external, 16-bit internal
e. 16-bit external, 32-bit internal

___ 7. What is a pulse called that is used to synchronize a digital circuit?

a. Synchronizer
b. Timer
c. Pulser
d. Gate
e. Clock

___ 8. What internal component on the 8086/88 results in quick execution of instructions?

a. Address
b. Clock
c. Queue
d. Counter
e. Decoder

___ 9. Which describes address and data lines on the PC system bus?

a. Multiplexed
b. On separate pin locations
c. Latched and repowered
d. Both a and b
e. Both b and c

___ 10. Which describes address and data lines on the 8086/88 microprocessor?

a. Multiplexed
b. On separate pin locations
c. Latched and repowered
d. Both a and b
e. Both b and c

___ 11. Which describes an I/O port address?

a. Unique
b. Only in hexadecimal numbers
c. Only in binary numbers
d. Repeated on 64K boundaries
e. All of the above

___ 12. Why are bus signals to the breadboard buffered?

a. The PC can only supply about two TTL fan outs per slot.
b. Shorts on the breadboard are isolated from the PC bus.
c. Breadboard signals must be inverted.
d. Both a and b
e. Both b and c

___ 13. Both high and low address bus and control bus logic levels are applied to logic gates in a predetermined pattern to select a memory chip. What is this process called?

a. Encoder
b. Decoder
c. Reset
d. Queue
e. Read

Match the following terms and descriptors.

____	14.	64K	a.	Construct circuit
____	15.	Enable	b.	Make signal lines accessible for test
____	16.	Buffer	c.	Make active
____	17.	Kludge	d.	Respond to one of many
____	18.	Breakout box	e.	Eight binary combinations
____	19.	Three-bit code	ab.	To separate
____	20.	Decoder	ac.	65,535

ANSWERS TO UNIT TEST

1.	d	(I9)
2.	c	(I9)
3.	a	(I9)
4.	c	(I9)
5.	a	(I9)
6.	b	(I9)
7.	e	(I9)
8.	c	(I9)
9.	e	(I9)
10.	a	(I9)
11.	a	(I7)
12.	d	(I7)
13.	b	(I7)
14.	ac	(I7)
15.	c	(I7)
16.	ab	(I7)
17.	a	(I7)
18.	b	(H7-8)
19.	e	(I7)
20.	d	(I7)

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS
UNIT VI: INTERFACE AND DATA TRANSFER TECHNIQUES

UNIT OBJECTIVE

Upon completion of this unit, the student should be able to construct and program simple interface circuits. This knowledge will be evidenced by achieving the specified score on the job sheets and unit test.

SPECIFIC OBJECTIVES

After completion of the lessons in this unit, the student should be able to:

1.1 Define terms and definitions.

1.2 Explain the operation of a PPI (Programmable Peripheral Interface).

1.3 Construct the I/O port (competency I7 and I8, JS 1.1)

1.4 Program the PPI I/O interface in BASIC. (competency I7, JS 1.2)

1.5 Program the PPI I/O interface in Assembly. (competency I7, JS 1.3)

1.6 Solve traffic light problem. (competency I7, JS 1.4)

2.1 Explain common technical terms related to data communications.

2.2 List three common data communications standards.

2.3 Explain error detection purposes.

2.4 Differentiate between types of carriers.

2.5 Differentiate between common interfaces.

2.6 Identify types of networking.

2.7 Identify types of protocol. (competency I8)

2.8 Construct a null-modem. (competency I7, JS 2.1)

2.9 Differentiate between asynchronous and synchronous methods of moving information.

2.10 Describe tests used when testing data communication equipment.

2.11 Identify types of data test equipment.

2.12 Differentiate between the kinds of programs available through bulletin boards. (competency I8)

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT VI: INTERFACE AND DATA TRANSFER TECHNIQUES

CONTENTS OF THIS UNIT

A. Objective sheet

B. References

C. Notes to the instructor

D. Lesson plans

1. Lesson 1: THE 8255A PPI

   a. Information outline

   b. Transparency masters

      TM 1.1: 8255A PPI chip
      TM 1.2: PPI block diagram

   c. Handouts

      HO 1.1: PPI I/O port schematic
      HO 1.2: Decoder and PPI schematic

   d. Job sheets

      JS 1.1: Construct the I/O port
      JS 1.2: Program the PPI I/O interface in BASIC
      JS 1.3: Program the PPI I/O interface in Assembly
      JS 1.4: Solve traffic light problem

2. Lesson 2: DATA COMMUNICATIONS

   a. Information outline

   b. Transparency master

      TM 2.1: Types of communication lines

   c. Handouts

      HO 2.1: EBCDIC code
      HO 2.2: Interface connectors

   d. Job sheets

      JS 2.1: Construct null-modem

E. Unit test

F. Answers to unit test

REFERENCES

A. Books

1. Brey, Barry B. 8086/8088 Microprocessor: Architecture, Programming, and Interfacing. Columbus, OH: Merrill, 1987.

2. Brey, Barry B. The Z80 Microprocessor: Hardware, Software Programming and Interfacing. Englewood Cliffs, NJ: Prentice-Hall, 1988.

3. Coffron, James W. Microprocessor Programming, Troubleshooting, and Interfacing: The Z80, 8080, and 8085. Englewood Cliffs, NJ: Prentice Hall, 1988.

4. Dailey, Denton J. Small Computer Theory and Applications. New York: McGraw-Hill, 1988.

5. Gaonker, Ramesh. The Z80 Microprocessor: Architecture, Interfacing, Programming, and Design. Columbus, OH: Merrill, 1988.

6. Gillmore, Charles M. Microprocessors: Principles and Applications. New York: McGraw-Hill, 1989.

7. Hall, Douglas V. Microprocessors and Interfacing: Programming and Hardware. New York: McGraw-Hill, 1986.

8. Harman, Thomas L. The Motorola MC68020 and 68030 Microprocessors: Assembly Language, Interfacing, and Design. Englewood Cliffs, NJ: Prentice-Hall, 1989.

9. Hughes, Fredrick W. Microprocessor Technology: Theory and Experimentation. Englewood Cliffs, NJ: Prentice-Hall, 1989.

10. Kersey, Roger M. Personal Computer Operation and Troubleshooting. Englewood Cliffs, NJ: Prentice Hall, 1989.

11. Lafore, Robert. Assembly Language Primer for the IBM PC & XT. New York: The Waite Group, 1984.

12. Miller, Michael A. The 68000 Microprocessor: Architecture, Programming, and Applications. Columbus, OH: Merrill, 1988.

13. Norton, Peter. Inside the IBM PC. New York: Brady, 1986.

14. O'Connor, Patrick Joseph. Digital and Microprocessor Technology. 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1989.

15. Pasahow, Edward J. Microprocessor Technology and Microcomputers. New York: McGraw-Hill, 1988.

16. Tocci, Ronald J., and Lester P. Laskowski. Microprocessors and Microcomputers: Hardware and Software. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1987.

B. Curriculum guides

1. Brown, A. O. Advanced Microcomputer Service Technician. Stillwater, OK: Mid-America Vocational Curriculum Consortium, 1985.

2. Vorderstrasse, Ron, and Leo Siebert. Communications Electronics. Stillwater, OK: Mid-America Vocational Curriculum Consortium, 1988.

NOTES TO THE INSTRUCTOR

A. Suggested approaches to teaching this unit

1. Demonstrate each interface circuit.
2. Demonstrate serial and parallel interface devices, such as printers.
3. Explain that controlling the PPI demonstrates how industrial controls are implemented.
4. Establish communications with modem and bulletin boards (BBS).

B. Suggested student activities

1. Research additional interface circuits.
2. Experiment with controlling small DC motors and relays with the PPI I.C. (Lesson 1).
3. Using a modem and bulletin boards (BBS), report to other students about BBS activities and services.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT VI: INTERFACE AND DATA TRANSFER TECHNIQUES

LESSON 1: THE 8255A PPI

I. Terms and definitions

A. Hard-wired--circuit wired specifically for one purpose; must be rewired to reconfigure the circuit

B. Interface--to connect two or more things together; a common meeting point

C. PPI--Programmable Peripheral Interface I.C., one of the many programmable support chips used to simplify interface design. The "smart" chip can be reconfigured (for example, from input to output) under program control without rewiring the circuit. Programmable interface adapters relieve the CPU and the programmer from much detailed instruction code.

II. Hardware interface

A. PPI

1. Hardware interface to the microcomputer can be done with conventional TTL logic chips.

2. All of the TTL interface circuitry must be hard-wired to a specific input or output configuration.

a. Reduces flexibility

b. Makes changes difficult

3. Each microprocessor family has several programmable devices to easily implement I/O and relieve the microprocessor from some of the work load.

4. Devices are available for parallel I/O, serial I/O and timing functions.

5. In the Intel family, the 8255A is a programmable peripheral interface (PPI) LSI chip designed to easily implement parallel I/O to the 8086/88.

6. The 8255A provides a flexible parallel interface.

NOTE: Any of the following can be selected under software control.

a. One-bit input and output ports

b. Four-bit input and output ports

c. Eight-bit input and output ports

d. Strobed inputs or outputs

e. Strobed bidirectional input/outputs

f. Level-sensitive inputs

g. Latched outputs

B. Input side **(TM 1.1)**

1. The input side of the 8255A is the microprocessor interface.
2. Data is transferred over an eight-bit bidirectional data bus D0-D7.
3. When the CPU performs an input or output bus cycle to a register address within the device, commands, status information and data are transferred over the data bus between the 8086/88 and the 8255A.
4. The read/write signals control timing of the data transfers to and from the 8255A.
5. The source or destination register within the 8255A **(TM 1.2)** is selected by one of the following.
   a. Two-bit register select code A1 and A0
   b. Port A register selected by A1A0 = 00
   c. Port B selected by A1A0 = 01
   d. Port C selected by A1A0 = 10
   e. Internal control register selected by A1 A0 = 11
6. The chip select ($\overline{CS}$) pin must be a logic 0 to enable the 8255A for input or output operations during a read or write operation.
7. Chip select is connected to the address decoding circuitry.
8. A logic 0 on RESET (during microprocessor power up) initializes the 8255A, clears internal registers and configures all I/O ports for input mode.

C. Output side **(TM 1.1)**

1. The output side of the 8255A has a total of 24 I/O lines.
   a. Port A has eight I/O lines (PA0-PA7).
   b. Port B has eight I/O lines (PB00-PB7).
   c. Port C has two groups of four I/O lines (PC0-PC7).
2. The control register is represented on the block diagram **(TM 1.2)** by Group A and Group B control blocks.
3. The control register is write only.
4. The control register is controlled by software and configures each port for input or output.
5. The control register is selected by register code A1 A0 = 11. Then, a control word is stored in the register.
6. Within the register, bits D0-D2 correspond to the group B control block on the block diagram.

7. Bit D0 configures the lower nibble of port C for input or output operation.

   a. Set bit D0 for an input operation.

   b. Reset the bit for output operation.

8. Bit D1 configures port B as a byte-wide input or output port.

   a. Set bit D0 for an input operation.

   b. Reset the bit for output operation.

9. Bit D2 is the mode select bit for port B and the lower nibble of port C.

   a. A logic low in bit D2 selects mode 0.

   b. A logic high in bit D2 selects mode 1.

10. Bits D3-D6 within the control register correspond to the group A control block in the block diagram.

11. Bit D3 configures the upper nibble of port C.

   a. Logic 1 selects input operation.

   b. Logic 0 selects output operation.

12. Bit D4 configures port A.

   a. Logic high selects input.

   b. Logic low selects output operation.

13. Bits D5 and D6 select group A's mode of operation (mode 0, 1 or 2)

14. Bit D7 is the mode set flag. A logic 1 changes operation mode.

D. Mode 0

1. Mode 0 selects simple I/O operation. Port lines can be configured as level-sensitive inputs and latched outputs.

2. To set all ports to mode 0, load bit D7 with logic 1 and set the mode set flag.

3. To set group B to mode 0, load bit D2 with logic 0.

4. To set group A to mode 0, load bits D6 and D5 with logic 0.

5. Port A and port B can now be configured as eight-bit input or output ports by setting or resetting bits D4 and D1.

6. Port C can be configured as two independent four-bit input or output ports by setting or resetting bits D3 and D0.

7. For example, consider the following control words. Control word 1000 0000 would select mode 0.

   a. Configure ports A and B as byte-wide output ports.

b. Configure port C as a byte-wide (both upper and lower nibbles) output port.

8. Control word 1000 0011 would select mode 0 and configure as output port A and the upper nibble of port C. It would also configure port B and the lower nibble of port C as input ports.

9. Combinations of the four control bits D4, D3, D1 and D0 will select any one of 16 different mode 0 I/O configurations.

10. The data ports in mode 0 are unidirectional, that is, they must be either input or output at any given time.

E. Mode 1

1. Mode 1 operation is known as strobed I/O.

2. Select mode 1.

a. Control bit D7 = 1 to activate the mode set flag

b. Control bit D6 = 0

c. Control bit D5 = 1

d. Control bit D2 = 1

3. In mode 1, port A is configured as a byte-wide I/O port with port C configured as lower-nibble, four-bit control/data port.

a. Port B is configured as a byte-wide I/O port with port C configured as a upper-nibble, four-bit control/data port.

b. The four-bit ports are handshake signals.

4. Data applied to an input port is strobed in with a signal produced by the external device hardware.

5. When used as an output port, handshake signals are provided that indicate when new data are available at the output and when an external device has read the outputs.

6. The data ports in mode 1 are unidirectional--they must be either input or output at any given time.

F. Mode 2

1. Mode 2 is known as strobed bidirectional I/O.

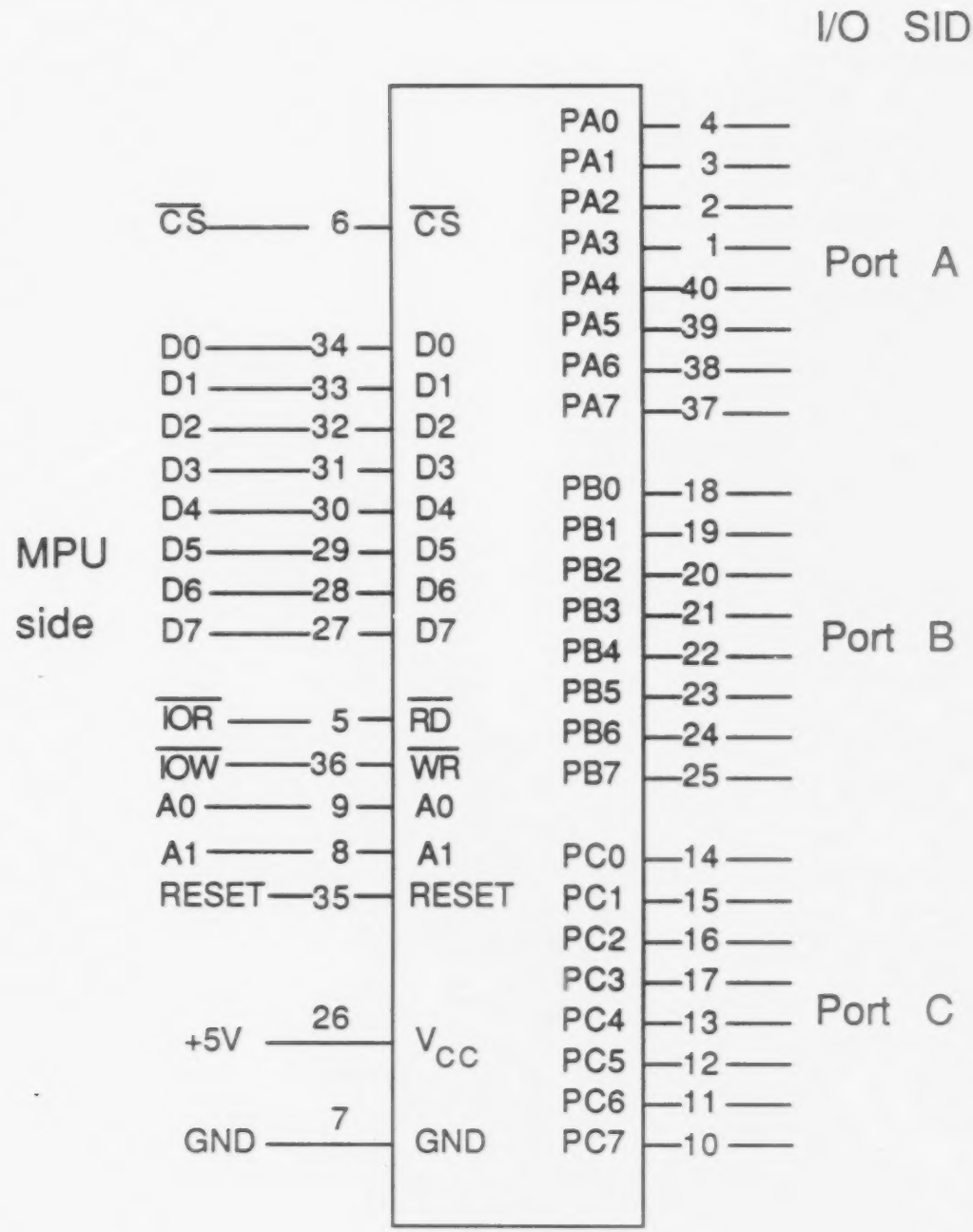
2. The difference is that the port now works as both input and output. Control signals are provided for both input and output.

3. Note that only port A is capable of mode 2 operation.

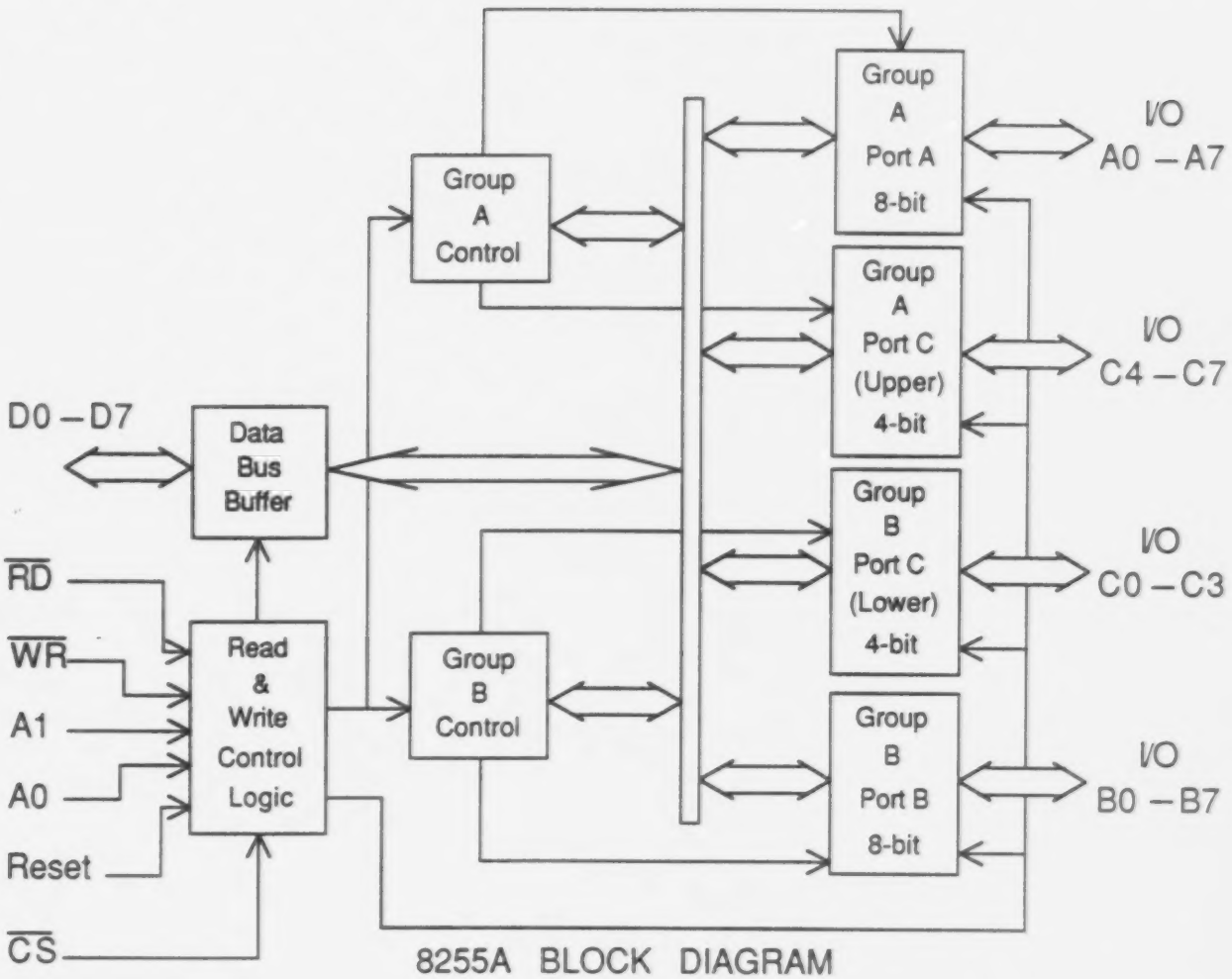
4. In practice, A and B ports are often configured for different modes of operation.

Example: Port A as mode 2, port B as mode 0

8255A PPI CHIP

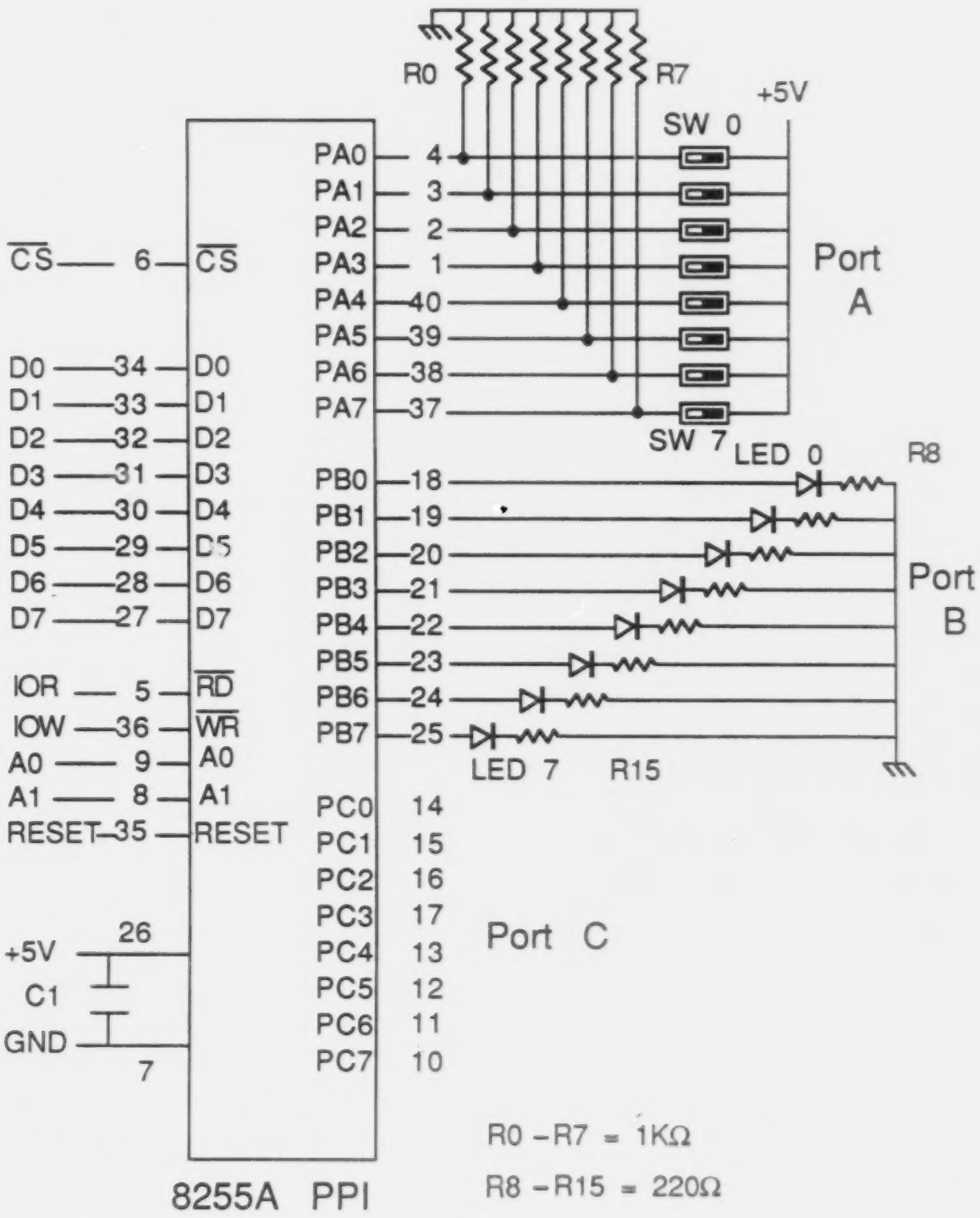


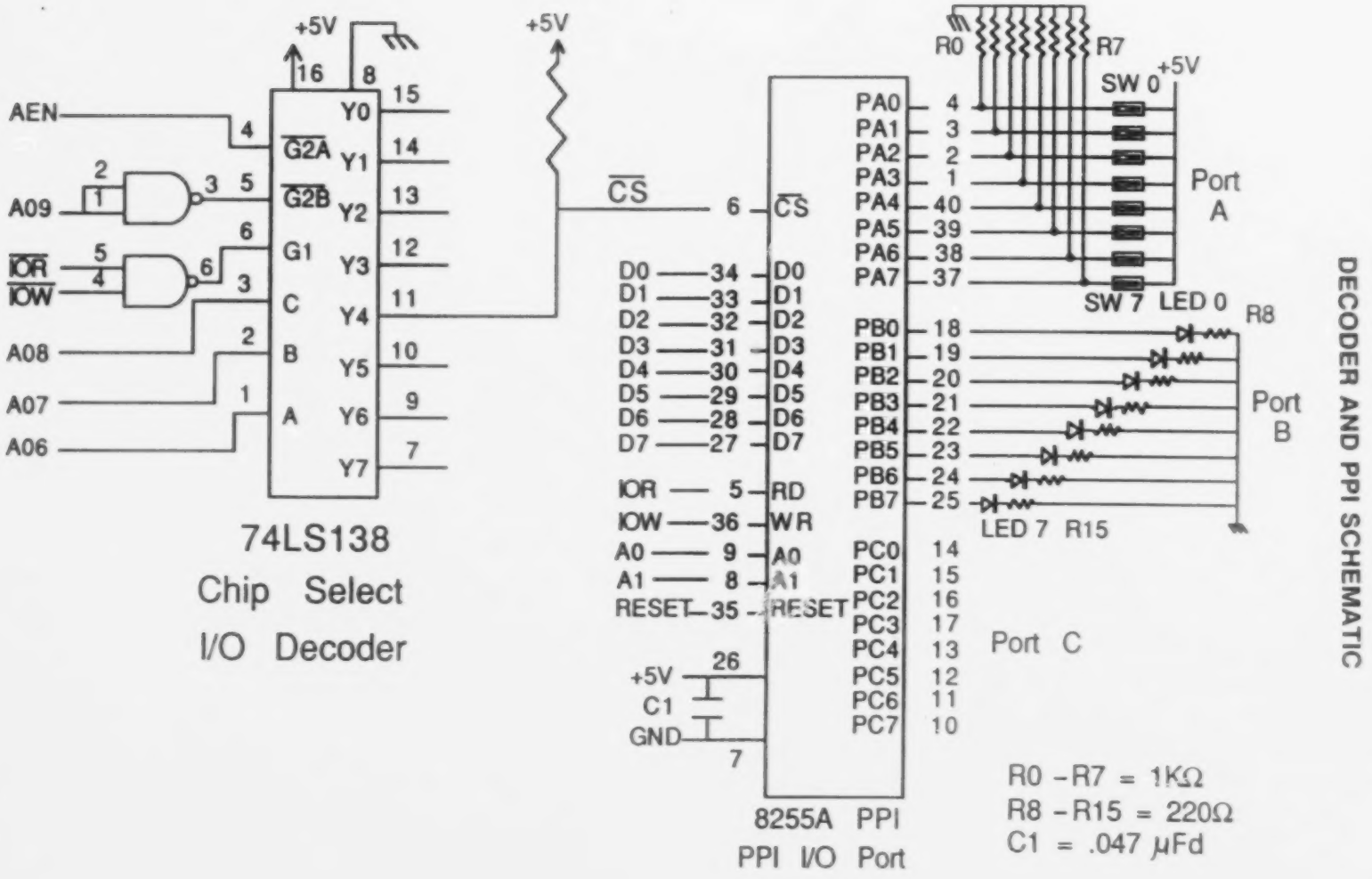
PPI BLOCK DIAGRAM



8255A BLOCK DIAGRAM

PPI I/O PORT SCHEMATIC





Breadboard I/O Port

CONSTRUCT THE I/O PORT

Equipment:

HO 1.1: PPI I/O port schematic
HO 1.2: Decoder and PPI schematic
8255 PPI
8 LEDs
8 1K resistors
8 220Ω resistors
Eight-position DIP switch
DVOM

Procedure 1: Construct PPI port (HO 1.1)

		Evaluation 3	2	1	N
1.	Keep wires neat and close to the breadboard. Connect a .047 disk ceramic bypass capacitor between the ground bus and the +5V bus.	—	—	—	—
2.	Examine the 8255A schematic. The lines marked BD0 through I/O EN are on the left of the schematic. They are the bus lines or buffered lines if using a buffer/decoder card. Mount the 8255A so that it straddles the center breadboard channel to the right of the address decoder circuit. Place the clocking mark (dimple, depression or dot) on the I.C. towards the top of the breadboard.	—	—	—	—
3.	Pin 1 will then be the top left pin, and pin 40 will be the top right pin. Position the 8255A so the bottom of the chip is near the vertical center of the block. Connect a black wire from the ground bus to pin 7 and a red wire from the +5V bus to pin 26 on the 8255A.	—	—	—	—
4.	Mount the eight-position DIP switch straddling the center channel on the breadboard near the top of the breadboard area. As much as possible, keep the right half of the breadboard area free for future wiring. Connect a hook-up wire from each of the buffered bus lines to the corresponding pin on the 8255A.	—	—	—	—
5.	Cut wires to length, strip ends and turn a 90° bend at the ends with the needle-nose pliers. Keep wires neat and close to the board.	—	—	—	—
	NOTE: Do not cross wires over any I.C. Avoid a "rats nest" appearance. Alternate colors of wire (not red or black) for easier circuit tracing.				
6.	Port A of the 8255A will be used as an input port. Connect a DIP switch to each corresponding data line, PA0-PA7. Switch 1 goes to PA0, switch 2 to PA1, etc.	—	—	—	—
	a. Note that each line has a 1KΩ resistor from the line to ground. Cut the resistor leads to length and turn a 90° bend at the wire ends.				
	b. Connect one end of the resistor to the ground bus and the other end to the switch terminal. Keep resistors straight horizontally.				

c. Bend eight short U-shaped jumpers of bare wire. Jump switch pin 9 to 10, 10 to 11, 11 to 12, etc., to connect pins 9-16 together. Then, run one red wire to the +5V bus. This will be neater than eight separate wires to the bus.

7. Port B of the 8255A will be used as an output port with an LED connected to each line, PB0-PB7. — — — —

a. Each line (PB0-PB7) connects to the anode of an LED. The cathode of the LED connects to a 220Ω resistor, and the resistor connects to ground. Find the cathode and anode pins of the LEDs.

NOTE: If the LED leads are clipped, be sure to keep the cathode lead about 1/4" shorter than the anode lead for future identification. Leads should be no shorter than 3/4" to prevent breakage.

b. Mount LEDs to straddle the center channel of the left breadboard. Line up the LEDs straight across from each corresponding pin of the 8255A with the anode end facing the I.C.

c. Cut the 220Ω resistors to length, turning the ends down 90° with the needle-nose pliers. The resistors will reach from the ground bus to each LED cathode.

d. Resistors and LEDs should be neat and evenly spaced.

8. Turn on the PC power to see if it will still boot. Power up the 8255A; check again to see that the PC operates normally. — — — —

9. Ask the instructor to check that the circuit was constructed correctly. — — — —

10. Was the circuit analyzed correctly? — — — —

11. Discuss the project with the instructor in a knowledgeable manner. — — — —

12. How long did it take to perform these steps? (_____ hrs., _____ min.) — — — —

13. Turn equipment off; store equipment and tools properly. — — — —

Procedure 2: Test decoder (HO 1.2)

1. Attach a logic probe to $\overline{CS}$ (chip select line) on the breadboard. The probe should show a binary 1 (high). — — — —

2. Run GWBASIC by typing GWBASIC at the DOS "A>" prompt. Enter "OUT &H300,1". This should cause the logic probe to blink to a binary 0 (low) once. If the probe did not show low, enter the BASIC command again. The low indicates that I/O port 300_H has been enabled. (If a low is not seen, consult the instructor.) — — — —

3. If chip select pulls low, check-out is complete. — — — —

4. Ask the instructor to check that the circuit is constructed correctly. — — — —

5. Was the circuit analyzed correctly? — — — —

6. Discuss the project with the instructor in a knowledgeable manner. — — — —

7.	How long did it take to perform these steps? (_____ hrs. _____ min.)	_	_	_	_
8.	Turn equipment off; store equipment and tools properly.	_	_	_	_
Average of above evaluations*		_	_	_	_

*This average is the student's final evaluation for competency I8 and partial evaluation for I7.

PROGRAM THE PPI I/O INTERFACE IN BASIC

Equipment:

HO 1.1: PPI I/O port schematic
HO 1.2: Decoder and PPI schematic
8255 PPI
8 LEDs
8 1K resistors
8 220Ω resistors
Eight-position DIP switch
DVOM

NOTE: Write a BASIC program to read the DIP switches at port A and write the switch settings to port B.

Procedure:	Evaluation 3	2	1	N
1. Have the program loop and continue to read port A and write port B for the following.	—	—	—	—
a. If any DIP switch at port A is a logic 1, then the corresponding bit at port B will light the LED.				
b. If any DIP switch at port A is a logic 0, then the corresponding bit at port B will turn the LED off.				
2. Ask the instructor to check that the circuit was constructed correctly.	—	—	—	—
3. Was the circuit analyzed correctly?	—	—	—	—
4. Discuss the project with the instructor in a knowledgeable manner.	—	—	—	—
5. How long did it take to perform these steps? (_____ hrs. _____ min.)	—	—	—	—
6. Turn equipment off; store equipment and tools properly.	—	—	—	—
Average of above evaluations*	—	—	—	—

*This average is the student's partial evaluation for competency I7: Describe interfacing techniques.

PROGRAM THE PPI I/O INTERFACE IN ASSEMBLY

Equipment:

HO 1.1: PPI I/O port schematic
HO 1.2: Decoder and PPI schematic
8255 PPI
8 LEDs
8 1K resistors
8 220Ω resistors
Eight-position DIP switch
DVOM

NOTE: Write an Assembly language program to read the DIP switches at port A and write the switch settings to port B.

Procedure:	Evaluation 3	2	1	N
1. Have the program loop and continue to read port A and write port B for the following.	—	—	—	—
a. If any DIP switch at port A is a logic 1, then the corresponding bit at port B will light the LED.				
b. If any DIP switch at port A is a logic 0, then the corresponding bit at port B will turn the LED off.				
2. Ask the instructor to check that the circuit was constructed correctly.	—	—	—	—
3. Was the circuit analyzed correctly?	—	—	—	—
4. Discuss the project with the instructor in a knowledgeable manner.	—	—	—	—
5. How long did it take to perform these steps? (_____ hrs. _____ min.)	—	—	—	—
6. Turn equipment off; store equipment and tools properly.	—	—	—	—
Average of above evaluations*	—	—	—	—

*This average is the student's partial evaluation for competency I7: Describe interfacing techniques.

SOLVE TRAFFIC LIGHT PROBLEM

Equipment:

HO 1.1: PPI I/O port schematic
HO 1.2: Decoder and PPI schematic
8255 PPI
8 LEDs
8 1K resistors
8 220Ω resistors
Eight-position DIP switch
DVOM

Procedure:	Evaluation 3	2	1	N
1. Draw a schematic diagram to connect four red LEDs, four yellow LEDs and four green LEDs in a traffic light arrangement. The LEDs will be in a north-south, east-west pattern as shown below.	—	—	—	—
2. Disconnect parts connected to the outside of the PPI used in the previous experiments.	—	—	—	—
3. The LED traffic lights will be wired to the I/O side of the 8255, so show pin connections of the 8255 to the LEDs. Indicate LED color on the schematic.	—	—	—	—
4. Breadboard the traffic lights connected to the 8255.	—	—	—	—
5. Write a program in your choice of languages to operate the traffic lights in a pattern similar to real traffic lights.	—	—	—	—
6. Show your operating traffic lights to the instructor. Was the circuit correctly constructed?	—	—	—	—
7. Ask the instructor to check that the circuit was constructed correctly.	—	—	—	—
8. Was the circuit analyzed correctly?	—	—	—	—
9. Discuss the project with the instructor in a knowledgeable manner.	—	—	—	—

NORTH

0
0
0

WEST 0 0 0 0 0 0 EAST

0
0
0

SOUTH

0 = LED

10. How long did it take to perform these steps? (_____ hrs. _____ min.) _ _ _ _

11. Turn equipment off; store equipment and tools properly. _ _ _ _

Average of above evaluations* _ _ _ _

*This average is the student's partial evaluation for competency I7: Describe interfacing techniques.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT VI: INTERFACE AND DATA TRANSFER TECHNIQUES

LESSON 2: DATA COMMUNICATIONS

NOTE: This lesson addresses common interfacing techniques, various networks and error detection systems.

I. Terms and definitions

A. Baud rate--total number of transmissions per second. If various sized bits are transmitted, the duration of the smallest bit must be used.

B. Bit rate--total number of information bits transmitted per second

C. Characters--specific number of ones and zeros

D. Clock pulse--pulse used to synchronize a digital circuit

E. Data--information which can be produced or processed by a computer

F. Data communications--technology of transferring data from one point to another between various distances

G. EBCDIC--Extended Binary Coded Decimal Interchange Code

H. LAN--Local Area Network; links a group of PCs together

I. Modem--modulator/demodulator, converts digital information to audio tones for transmission over telephone lines and audio tones back to digital information

J. Network--configuration of communication facilities that connect the user equipment to the TX/RX equipment

K. Parity--bit added to a code to give either an odd or even number of binary highs

L. TX/RX--TX means transmit or transmitter (sending) equipment; RX means receive or receiver equipment.

II. Common data codes and their characteristics

A. The first binary data codes for messages were bonfires, smoke signals, flags and mirrors.

B. Morse code was the first electrical data communications code. It uses dots and dashes for communications.

C. Baudot code was the first electrical binary code using either a signal or no signal. It was used primarily in Teletype machines.

D. BCD code (binary coded decimal) consists of only numeric characters. It is useful for calculations and storing information.

E. EBCDIC code is alphanumeric. **(HO 2.1)**

1. Contains an eight-bit code for each character

2. Used in many mini and mainframe computers

F. ASCII code is the most widely used data code internationally. (HO 1.1, Unit IV)

1. Contains a seven-bit code plus a one-bit parity

2. Alphanumeric

3. Used by personal computers, printers and CRT displays

III. Error detection methods

A. Reason for error detection

1. Electrical noise on the lines can garble the code.

2. A burst of static can be interpreted by the RX as a binary one.

3. The extra binary one could cause a control code to become the wrong code, an alpha code to print as the wrong letter, or a numeric code to print the wrong bill or paycheck.

B. Echo checks

1. Information sent from one station to another is sent back to the original station and then compared for errors.

2. Since other terminals cannot use the line while echo checks are performed, this is not a widely used detection method.

C. Parity checks

1. Either a one or zero is added to the end of a character, making an odd or even number of binary highs.

2. The receiver can check for this odd or even parity.

3. If bits received do not match the parity, an error has occurred.

4. The IBM PC uses a memory parity check.

a. Eight bits in a byte of memory

b. A ninth bit is used for parity

D. CRC (cyclic redundancy check)

1. A mathematical polynomial is used.

2. Bits are added, then divided by a constant.

3. The remainder on both TX and RX must compare.

4. CRC is the most accurate and widely used method of error detection.

5. Disk drives on personal computers use CRC.

IV. Types of carriers

A. Local carrier (common carrier)--the line connecting the user's equipment to the local telephone network

B. Long distance carrier (satellite carrier)--communication connections locally and nationwide

C. Value-added network--data carriers who provide an added service through the carriers

D. Resale carrier

1. A carrier who leases lines from other carriers

2. Resells time on leased lines to users

E. Radio carriers--carriers who provide service through radio links, such as cellular phones

V. Common interfaces

A. RS232 **(HO 2.2)**

1. Developed by EIA in conjunction with Bell systems

2. A standard serial interface for data communications equipment

3. Used by CRT terminals and serial printers

4. Uses 25-pin DB25 connector

a. Not all pins must be used.

b. The minimum interface uses pins 1, 2, 3 and 7.

B. V.24

1. Compatible with the RS232

2. Established to interface with circuits of foreign countries

C. RS449

1. This interface was developed to replace RS232.

2. It includes RS422 (a balanced line) and RS423 (an unbalanced line).

3. Due to LAN growth, the RS232 continued to be used.

D. IEEE-488--most accepted interface to connect data systems to test equipment and for scientific apparatus

E. Parallel interface (Centronics type) **(HO 2.2)**

1. Used in communicating digital information in parallel form

2. Standard for parallel printer connection in personal computers

VI. Basic types of networking

A. Packet switching

1. Groups of data called packets are sent as a complete packet.

2. Charges are only for time and distance used.

B. LAN

1. Over telephone lines between locations or cities

2. In houses or between offices

3. Popular PC networks for business or education

a. Novell network

b. IBM Token Ring

VII. Types of protocol

A. Line discipline **(TM 2.1)**

1. Types of communications lines

a. SPX--Simplex

b. HDX--Half duplex

c. FDX--Full-duplex

d. F/FDX--Full/full duplex

B. Handshaking

1. Process to establish a communications line

2. Types of lines used

a. Dial up--routing through whatever lines are available

b. Dedicated lines--privately leased lines for a specified time

VIII. Modems

A. Modulator--device that converts digital to analog signals

B. Demodulator--device that converts analog to digital signals

C. DSU--device that converts digital signals into pulse transmitted signals

D. Null-modem (RS232) **(HO 2.2)**

1. When a modem and telephone line are used to connect equipment together, the modem routes transmitted data on one end to received data on the other end.

2. When equipment must be connected to peripheral equipment and a modem is not used, a problem arises.

3. Note that on an RS232 connector, pin 2 is transmitted data and pin 3 receives data.

4. If two pieces of equipment are connected together with a normal RS232 cable, each piece would have transmit-to-transmit and receive-to-receive capabilities.

5. A null-modem cable is used to cross the transmit and receive connections.

   a. Connect pin 2 on one end to pin 3 on other end.

   b. Connect pin 3 on one end to pin 2 on other end.

6. The two pieces of equipment can now communicate.

IX. Methodology of moving information

A. Asynchronous

1. Transmission of characters at random

2. Uses start and stop pulses

B. Synchronous

1. Transmission of characters at specific times

2. Uses clock pulses

X. Methods of testing data communications equipment

A. Manual test--equipment manually connected by technician

B. Switching test--equipment previously connected through relays and switches to be activated by technician

C. Computerized test--testing through use of a computer operated by a technician

D. Computer-controlled test--preprogrammed computers used without a technician

XI. Types of data test equipment

A. Analog test equipment

1. Signal-level meter

2. Signal/noise ratio meter

3. Signal generator

4. Line monitors

B. Digital test equipment

1. BERT--bit error rate tester

2. CERT--character error rate tester

3. BLERT--block error rate tester

4. Breakout box

   a. Test instrument placed between the DTE and DCE to check or insert signals in the data path

   b. DTE--data terminal equipment

   c. DCE--data communications equipment

   d. Low-cost breakout boxes available for RS232 and Centronics

5. Loop-back adapter

   a. Output (TX) line attached to input (RX) line

   b. Allows equipment to "talk to itself" for test

XII. Bulletin boards

A. Hobby bulletin boards

1. Bulletin boards (BBS) networks are operated by individual operators (SYSOPS) for fun.

2. A computer with a large capacity hard disk is used as the host computer.

3. Individuals dial-up the telephone and modem attached to the BBS.

4. Software can be downloaded to the user or uploaded to the system operator.

   a. Public domain software is free for anyone to copy and use.

   b. Shareware software is not in the public domain.

      (1) Software is protected by copyright.

      (2) Users can freely copy the software to try it.

      (3) The user is expected to pay the author a fee if the software is used.

      (4) Shareware cannot be copied and sold by others.

      (5) A fee (up to about $9) can be made for the diskette and labor involved in making the copy, not for the software itself.

5. Commercial programs protected by copyright should never be copied.

6. A "chat" mode is usually provided so that two operators can send messages to each other (similar to a CB radio).

7. Electronic mail messages can also be left for other operators on the BBS.

B. Commercial BBS

1. Several national BBS networks are available, such as Compuserve, BIX and Dow-Jones.

2. A fee is charged for the time "logged on".

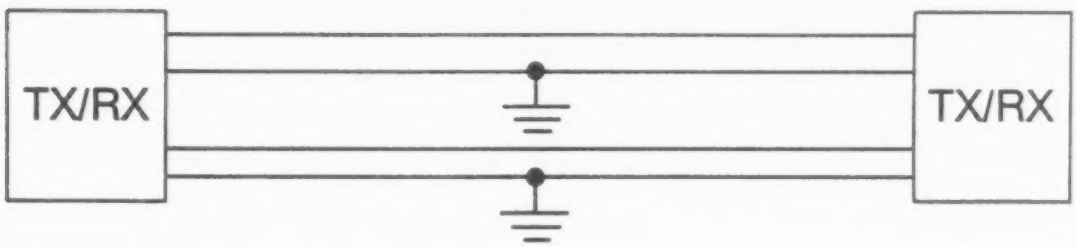
3. Since these bulletin boards are located in large cities, long-distance phone bills are involved.

4. Many of the same services available on local BBS are available on commercial BBS.

5. Extensive libraries of information are available.

TYPES OF COMMUNICATION LINES

SPX/HDX



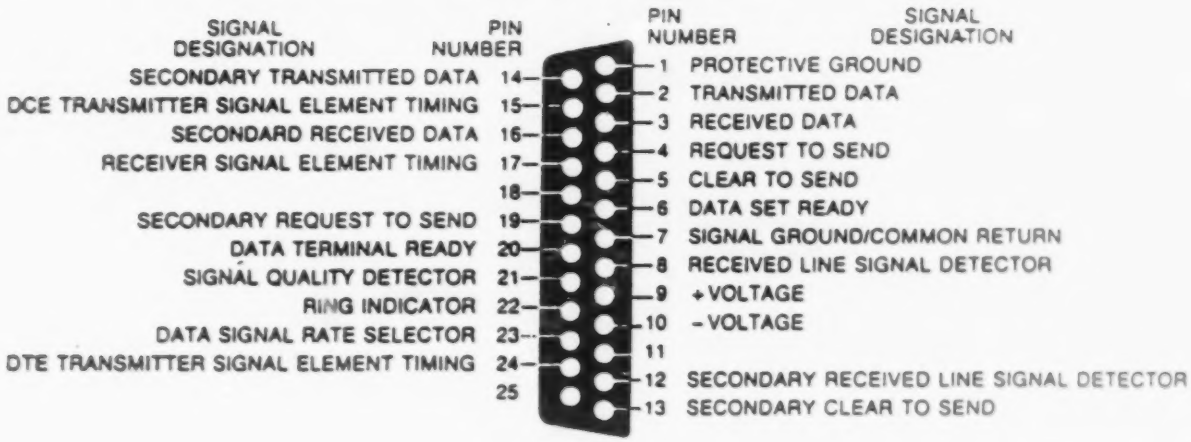
FDX

EBCDIC CODE

Bits			4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
			3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
			2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
			1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
8	7	6	5																
0	0	0	0	NUL	SOH	STX	ETX	PF	HT	LC	DEL			SMM	VT	FF	CR	SO	SI
0	0	0	1	DLE	DC_1	DC_2	DC_3	RES	NL	BS	IL	CAN	EM	CC		IFS	IGS	IRS	IUS
0	0	1	0	DS	SOS	FS		BYP	LF	EOB	PRE			SM			ENQ	ACK	BEL
0	0	1	1			SYN		PN	RS	UC	EOT					DC_4	NAK		SUB
0	1	0	0	SP										¢	.	>	(	+	|
0	1	0	1	&										!	$	*	)	;	
0	1	1	0	-	/										,	%	-	<	?
0	1	1	1											:	#	@	'	=	"
1	0	0	0		a	b	c	d	e	f	g	h	i						
1	0	0	1		j	k	l	m	n	o	p	q	r						
1	0	1	0			s	t	u	v	w	x	y	z						
1	0	1	1																
1	1	0	0		A	B	C	D	E	F	G	H	I						
1	1	0	1		J	K	L	M	N	O	P	Q	R						
1	1	1	0			S	T	U	V	W	X	Y	Z						
1	1	1	1	0	1	2	3	4	5	6	7	8	9						□

INTERFACE CONNECTORS

RS-232 Interface



Parallel Interface (Centronics type)

(R) INDICATES SIGNAL GROUND RETURN

Credit: Reproduced by permission, Black Box Corporation, Pittsburgh, Penn. © Copyright 1990, Black Box Corporation. All rights reserved.

CONSTRUCT STANDARD NULL-MODEM

Equipment:

DB25 connectors (2 each, male or female as required)
Length of connecting cable
Tools
VOM
RS232 breakout box

	Evaluation 3	2	1	N
Procedure 1: Determine cable requirements				
1. Locate two pieces of shop equipment that require a null-modem. (PC to serial printer, for example)	—	—	—	—
2. Examine the DB25 serial connectors on both pieces of equipment and determine which type of connector is needed to mate with the serial I/O ports.	—	—	—	—
How many male connectors are needed? _____ Female connectors? _____				
Procedure 2: Assemble cable				
1. Connect pin 1 of each connector to the cable protective ground shield.	—	—	—	—
2. Connect pin 7 (signal ground) of each connector with one of the cable wires.	—	—	—	—
3. Connect pin 2 (transmitted data) of plug 1 to pin three (received data) of plug 2.	—	—	—	—
4. Connect pin 3 (received data) of plug 1 to pin two (transmitted data) of plug 2.	—	—	—	—
5. Connect pin 20 (DTR) of plug 1 to pin 6 (DSR) of plug 2.	—	—	—	—
6. DTR signal is provided to indicate the power is on and that a line has been established. Connect pin 6 (DSR) of plug 1 to pin 20 (DTR) of plug 2.	—	—	—	—
7. Connect pins 4 (RST) and 5 (CTS) of plug 1 together.	—	—	—	—
8. Connect pin 4 of plug 1 to pin 8 (DCD) of plug 2.	—	—	—	—
9. When the data terminal set sends a RTS (request to send) it must receive a CTS (clear to send) indicating transmission is possible. At this time the terminal will receive a CTS (clear to send) signal at the DCT (data carrier detect) pin.	—	—	—	—

Null-modem connections

PLUG 1			PLUG 2
Shield	#1	#1	Shield
Ground	#7	#7	Ground
TX data	#2	#3	RX data
RX data	#3	#2	TX data
DSR	#6	#20	DTR
DTR	#20	#6	DSR
RTS	#4	#8	DCD
CTS	#5	#5	RTS
DCD	#8	#4	CTS

10. To complete the null-modem, connect pin eight of plug 1 to pins 4 and 5 of plug 2. _ _ _ _

Procedure 3: Test null-modem cable

1. Check continuity of cable with ohmmeter. _ _ _ _
2. Check for shorts between pins. Only pins 4 and 5 should be shorted together. _ _ _ _
3. Connect null-modem to equipment. Install RS232 breakout box in the serial line. _ _ _ _
4. Turn on equipment and test serial operation. Observe breakout box indicators for line activity. _ _ _ _
5. Show your operating traffic lights to the instructor. Was the circuit constructed correctly? _ _ _ _
6. Was the circuit analyzed correctly? _ _ _ _
7. Discuss the project with the instructor in a knowledgeable manner. _ _ _ _
8. How long did it take to perform these steps? (_____ hrs. _____ min.) _ _ _ _
9. Turn equipment off; store equipment and tools properly. _ _ _ _

Average of above evaluations* _ _ _ _

*This average is the student's final evaluation for competencies H7-8 and I7.

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT VI: INTERFACE AND DATA TRANSFER TECHNIQUES

UNIT TEST

Match the definition on the left with the term on the right.

____ 1.	First data communications code	a.	Baud rate
____ 2.	First code using either signal or no signal	b.	Baudot code
____ 3.	Pulse used to synchronize a digital circuit communications	c.	Bit
____ 4.	Total number of transmissions per second	d.	Bit rate
____ 5.	Technology of transferring data from one point to another between various distances	e.	Breakout box
		ab.	Centronics
____ 6.	Smallest element of information; one binary digit, either 1 or 0	ac.	Clock
____ 7.	Programmable support chip used to simplify interface design; can be reconfigured under program control without rewiring the circuit	ad.	Data
		ae.	Data communication
____ 8.	Test instrument placed between the DTE and the DCE to check or insert signals in the data path	bc.	Modem
____ 9.	Configuration of communication facilities that connects the user equipment to the TX/RX equipment	bd.	Morse code
		be.	Network
____ 10.	Standard serial communication interface for terminals and printers	cd.	PPI
____ 11.	Standard printer interface for personal computers	ce.	RS232
____ 12.	Device that converts analog-to-digital signals and digital-to-analog signals over telephone lines		

Select the option which answers the question or completes the statement.

____ 13. Which is an integrated circuit that can be programmed for either peripheral input or output functions?

a. CPU
b. PPI
c. MPU
d. ALU
e. I.P.

____ 14. For which are peripheral interface devices available?

a. Serial I/O
b. Parallel I/O
c. Timing functions
d. All of the above
e. None of the above

___ 15. How can the 8255A be configured?

a. One-bit input and output ports
b. Four-bit input and output ports
c. Eight-bit input and output ports
d. All of the above
e. None of the above

___ 16. A word is written to which of the following to initialize the 8255A?

a. Control register
b. Port A
c. Port B
d. Port C
e. All of the above

___ 17. Select the true statement about the PPI.

a. Input ports are latched.
b. Output ports are latched.
c. The 8255A is a hard-wired I/O port.
d. Output ports are level-sensitive.
e. There are eight output bits available.

___ 18. Which is <u>not</u> an error detection purpose?

a. Properly convert to another language
b. Find garbled code due to electrical noise on the lines
c. Eliminate misinterpreted binary 1s resulting from a burst of static
d. Compare station-to-station signals for errors
e. Perform echo checks

___ 19. Which type of carrier line connects the user's equipment to the local telephone network?

a. Common carrier
b. Satellite carrier
c. Value-added network
d. Resale carrier
e. Radio carrier

___ 20. Which interface is standard for printer connections in personal computers?

a. RS232
b. V.24
c. RS449
d. IEEE-488
e. Parallel

Identify true statements with an "a" and false ones with a "b".

___ 21. The Novell network is an example of a popular PC network in scientific research.

___ 22. Dedicated lines are those that are privately leased for a specified time.

___ 23. Synchronous describes the transmission of characters at specific times using clock pulses.

___ 24. When testing data communications equipment, a switching test involves a technician activating equipment previously connected through relays and switches.

____ 25. A BERT is a block error rate tester.

____ 26. Shareware software is free for anyone to copy and use.

27. List three common interface standards used with data communications equipment.

ANSWERS TO UNIT TEST

1.	bd	(I8)	14.	d	(I8)
2.	b	(I8)	15.	d	(I8)
3.	ac	(I8)	16.	a	(I8)
4.	d	(I8)	17.	b	(I8)
5.	ae	(I8)	18.	a	(I8)
6.	c		19.	a	(I8)
7.	cd	(I7)	20.	e	(I8)
8.	e	(H7-8)	21.	b	(I8)
9.	be	(I8)	22.	a	(I8)
10.	ce	(I8)	23.	a	(I8)
11.	ab	(I8)	24.	a	(I8)
12.	bc	(I8)	25.	b	(I8)
13.	b	(I8)	26.	b	(I8)

27. Any three of the following: (I7)
    a. RS232
    b. Centronics
    c. V.24
    d. RS449
    e. IEEE-488

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT VII: COMPUTER TROUBLESHOOTING

UNIT OBJECTIVE

Upon completion of this unit, the student should be able to diagnose, analyze, troubleshoot and repair a PC. This knowledge will be evidenced by completion of the job sheets to specified score.

SPECIFIC OBJECTIVES

After completion of the lessons in this unit the student should be able to:

1.1 Identify IBM PC hardware. (JS 1.1)

1.2 Tear down, reassemble and test a PC to check for normal operation. (JS 1.2)

1.3 Troubleshoot and repair a defective PC. (JS 1.3)

1.4 Test and replace a motherboard. (JS 1.1 and 1.2)

1.5 Replace a power supply. (JS 1.2)

1.6 Test and replace keyboards. (JS 1.1 and 1.2)

1.7 Test and replace disk drives. (JS 1.1 and 1.2)

1.8 Test and replace video displays and adapter cards. (competency I9, JS 1.1 and 1.2)

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT VII: COMPUTER TROUBLESHOOTING

CONTENTS OF THIS UNIT

A. Objective sheet

B. References

C. Notes to the instructor

D. Lesson plans

1. Lesson 1: PC TROUBLESHOOTING AND REPAIR

   a. Information outline

   b. Handouts

      HO 1.1: Beep indicators
      HO 1.2: Memory chip failure error codes
      HO 1.3: IBM DIP switch settings

   c. Job sheets

      JS 1.1: Run IBM advanced diagnostics
      JS 1.2: Tear down, reassemble and test a PC
      JS 1.3: Troubleshoot and repair a defective PC

REFERENCES

A. Books

1. Brenner, Robert C. IBM PC Advanced Troubleshooting and Repair. Indianapolis, IN: Howard W. Sams & Co. Inc., 1988.

2. Coffron, James W. Microprocessor Programming, Troubleshooting, and Interfacing: The Z80, 8080, and 8085. Englewood Cliffs, NJ: Prentice Hall, 1988.

3. Kersey, Roger M. Personal Computer Operation and Troubleshooting. Englewood Cliffs, NJ: Prentice Hall, 1989.

4. Lafore, Robert. Assembly Language Primer for the IBM PC & XT. New York: The Waite Group, 1984.

5. Norton, Peter. Inside the IBM PC. New York: Brady, 1986.

6. "Sams Computerfacts." Indianapolis, IN: Howard W. Sams & Co. Inc.

   a. Monitor: IBM Model 5151, IBM Model 5153 (CMT4). 1984.

   b. Computer: IBM PC 5150 (CSCS2). 1984.

c. Printer: IBM 5152-002 (CP3). 1984.

d. Printer: Epson MX-80 IIIF/T (CP1). 1984.

e. Computer: Apple II, II Plus (CC1). 1984.

f. Computer: Apple IIe, Model A2S2064 (CC10). 1985.

g. Computer: Apple IIc, Computer Model A2S4000, Monitor Model A2M4090 (CSCS7). 1985

7. IBM PC Hardware Maintenance series. Boca Raton, FL: International Business Machines Corporation. n.d.

a. IBM PC Hardware Maintenance and Service (6025072).

b. IBM PC Hardware Maintenance Service Personal Computer Systems (6280087).

c. IBM PC Hardware Maintenance Service (1502561).

d. IBM PC Hardware Maintenance Reference (01F0200).

B. Curriculum guide

1. Brown, A. O. Advanced Microcomputer Service Technician. Stillwater, OK: Mid-America Vocational Curriculum Consortium, 1985.

NOTES TO THE INSTRUCTOR

A. Suggested approaches to teaching this unit

1. Demonstrate the running and interpreting of diagnostics.

2. Discuss safety and ESD procedures.

3. Demonstrate computer troubleshooting techniques.

4. Demonstrate tear-down and reassembly of a PC.

5. Bug computers differently for JS 1.3 performance. Some suggestions follow.

a. Unplug the keyboard.

b. Turn power off, then on again quickly to place power supply in shut-down mode.

c. Connect CRT display to the wrong video adapter card.

d. Unplug power supply connector at motherboard.

e. Unplug power plug to disk drive.

f. Change speed of disk drive.

g. Pry memory chip out of socket, bend one pin out and replace chip.

CAUTION: Use extreme care! Chips can be irreversibly damaged!

h. Set switch block switches to wrong configuration.

i. Unplug loudspeaker.

j. Unplug signal cable at FDC connector.

k. Erase working copy of advanced diagnostic disk.

l. Place cellophane tape over a few pins on video adapter card.

m. Place cellophane tape over a few pins on FDC adapter card.

6. Use a copy of JS 1.3 and a defective computer for final testing.

B. Suggested student activities

1. Practice running diagnostics available in the shop.

2. Practice troubleshooting computers.

3. Have students do JS 1.3 as a final troubleshooting performance test.

C. Sources for diagnostics

1. Service diagnostics for PC system and disk drives

SuperSoft, Inc.
P.O. Box 1628
Champaign, IL 61820

2. Disk drive analyzer diagnostics

J & M Systems, Ltd.
15100-A Central S.E.
Albuquerque, NM 87123

MODULE: MICROPROCESSOR AND COMPUTER SYSTEMS

UNIT VII: COMPUTER TROUBLESHOOTING

LESSON 1: PC TROUBLESHOOTING AND REPAIR

NOTE: IBM's original personal computer with five expansion slots is a PC. The XT is an improved version of the PC with eight expansion slots. A compatible computer or a PC/XT clone will run software written for PC/XTs; also, most hardware is directly interchangeable with PC/XTs.

I. IBM PC hardware

A. The IBM PC/XT and compatible clone computer systems are made up of three basic units: the system unit or box, keyboard and cable, and CRT display.

1. A fourth unit, the printer, is usually attached to the computer system.

2. To troubleshoot and repair the PC system, the technician must know what makes up each unit or module.

B. System unit is the system box containing most of the computer modules and circuitry.

1. Motherboard or main circuit board

a. 8088 microprocessor

b. 8087 math coprocessor, if present

c. Memory I.C. chips

d. Clock and most computer circuits

e. Configuration switches

2. Power supply

a. Furnishes DC power to motherboard and disk drives

b. Cooling fan

c. Power supply--rated at 65 watts in older PC units and up to 150 watts in XTs and clones

3. Adapter cards that can be plugged into expansion slots

NOTE: The older PC has five slots for adapter cards, while the XT and most clones have eight slots for adapter cards.

a. Monochrome video adapter card with printer port

b. Color graphics video adapter card

c. Printer port adapter card

d. Floppy disk controller (FDC) adapter card

e. Hard disk controller adapter card

f. Clock/calendar adapter card

g. Internal modem adapter card

h. Game port adapter card

i. Serial RS232 adapter card

j. Bus mouse adapter card

k. Additional memory adapter card

l. Multi-function cards (combinations of above cards)

m. Other specialized cards

4. Loudspeaker

5. Floppy disk drives

6. Hard disk drive

NOTE: A computer with a hard drive usually also has one--and sometimes two--floppy drives.

C. The keyboard and coiled cord plugs into the system box.

1. The keyboard has its own microprocessor.

2. DC power for the keyboard comes from the system box.

D. The CRT monitor and video adapter card must be of the same type.

1. A monochrome TTL CRT must be used with a monochrome video adapter with printer port.

a. No graphics available with this combination

b. Designed to give sharp 80-column text for word processing

c. Monochrome CRT monitor incompatible with a color graphics card

d. Color CRT monitor incompatible with a monochrome card

2. The CGA color CRT must be used with the color graphics video adapter card.

a. CGA color must have RGB/I video from video adapter card.

(1) R = red video line
(2) G = green video line
(3) B = blue video line
(4) I = intensity or brightness line

b. A separate printer port adapter card must be used with the CGA color adapter card

c. Exception: Some imported color cards do have a printer port included.

d. Composite video is available from many CGA color graphics cards.

(1) Composite color CRT monitors
(2) Composite monochrome video monitors

3. EGA and VGA high-resolution color CRTs must use the correct EGA or VGA video adapter card.

4. Monochrome video graphics are possible with third-party monochrome video adapter cards.

a. Hercules graphics cards are the standard.

b. Clones of the Hercules are available.

c. The IBM TTL CRT can be used with Hercules cards.

d. Graphics resolution is comparable to EGA, but no color is available.

E. PCs have several printer options.

1. The most common printer for the IBM PC is the dot matrix connected to a parallel printer port. Letter-quality daisy wheel printers are also used.

2. These printers and plotters usually have RS232 serial interface rather than parallel.

3. Other options include laser printers and plotters for drafting applications.

II. Normal system operation

A. POST

1. When the PC is turned on, it performs pre-operation self tests (POST) to check for system failures.

2. If everything tests OK, the red light on drive A will turn on, then off, and the loudspeaker will sound one short beep.

3. The light and beep are signals that the computer is ready to operate.

4. Any other signals mean a problem has been found.

a. An error code will display on CRT screen.

b. The computer will refuse to operate.

B. Boot up

1. The microprocessor then starts disk drive A (or C if a hard drive is installed).

2. The computer looks for additional more advanced instructions on the diskette or batch file.

3. The operating system boots.

4. CRT display becomes normal brightness and displays the prompt.

C. Testing the PC

1. Become very familiar with normal operation of the PC. Test the PC in a systematic manner. Know what normal operation looks and sounds like.

2. If diagnostics are not available, the PC can be tested in the following manner.

   a. Give directory display command.

   b. Copy a file from one disk to another disk.

   c. Delete a file on the disk.

   d. Format a diskette.

   e. Run a familiar application program.

   f. If there is more than one floppy drive, repeat a-e for each drive.

   g. Dump screen text to printer (Shift-Prt Sc).

   h. Turn the computer off and remove the diskette.

   i. Insert boot disk, then turn computer on.

   j. Boot computer 2-3 times, repeating h and i.

3. If diagnostics are available, they will usually give a more thorough test.

D. Diagnostics

1. Advanced diagnostics diskettes are available from IBM for testing IBM computers.

2. The IBM maintenance diagnostic package includes an advanced diagnostic diskette and a manual with maintenance analysis procedures (MAPS).

3. Different advanced diagnostics are required for different models (PC, XT, AT, PS/2 Model 25, PS/2 Model 30, PS/2 Model 40 and 60).

4. Other manufacturers have comparable diagnostics for their computers.

5. Disk drive diagnostics are available from several sources to test disk drive alignment.

6. Diagnostics on both diskette and ROM are available from SuperSoft.

7. "Scratch" (formatted blank) diskettes are required when testing disk drives.

8. IBM advanced diagnostics have their own operating system; when making backup copies, DISKCOPY must be used. Do not use FORMAT and COPY to make the backups.

9. Boot up from the advanced diagnostic diskette and follow the menus.

   a. Observe normal operation of the diagnostic.

   b. If a manual is included, follow instructions in it.

III. System failures

A. Error codes warn of problems.

1. A failure detected during POST results in a beep indicator and/or an error code display in the upper left corner of CRT display.

2. The error code display may disappear after a few seconds; close observation may be required.

3. Beep indicators can conflict with error codes; ignore beeps and troubleshoot the error codes.

4. Replace the suspect major unit (FRU) with a new or known good unit.

5. An identical working PC is a common source of "swap out" units.

WARNING: There is a chance that a bad component could blow the systems!

B. Beeps from the loudspeaker can be the technician's guide to a major section of the PC system.

Beep indicator	Look for trouble at
No beep, dead PC	AC power, power supply
Continuous beep	AC power, power supply
Series of short beeps	System board
One long beep, one short one	System board
One long beep and two shorts	Display
One short beep, blank display, incorrect display	Display
One short beep, no disk boot, cassette BASIC	Diskette, disk drive

C. System error codes appear alone or with other numbers in upper left corner of display.

1. xxxx indicates any hexadecimal number present.

Error code	Look for trouble at
101	System board (motherboard)
201	RAM failure
xxxx=201	Memory failure
PARITY CHECK x	(See D below for codes)
PARITY ERROR 1	Memory--try seating RAM chips
301	Keyboard or cable
401	Monochrome adapter card
501	Color/graphics adapter card
601-626	Disk drive adapter card, cable, disk drive
PARITY CHECK 1 (with no other codes)	Power supply

2. Replace the suspect major unit (FRU) with a new or known good unit.

WARNING: There is a chance that a bad component could blow the systems!

IV. Motherboard

A. Memory chip failure error codes

1. Memory chips are located in four banks of nine chips near the left front of the motherboard. The parity chip is near the edge of the motherboard.

2. Memory error codes have four hexadecimal digits.

3. There are two types of PC motherboards in use; check the markings on the edge of the motherboard near the memory chips.

   a. Older type marked 16/64KB (PC only)

   b. Newer type marked 64/256KB (PC or XT)

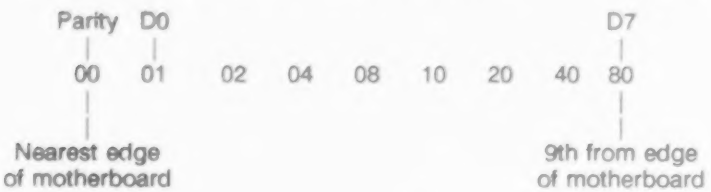
4. A defective memory chip is a common failure that is easily located and corrected.

B. PC motherboard

1. If the PC motherboard is marked 16/64KB, the first two characters of the error code indicate the failing bank of chips.

00	Bank 0	(Towards rear of PC)
04	Bank 1	
08	Bank 2	
0C	Bank 3	(Near front of PC)

2. The second two characters of the error code indicate the failing chip within the bank.



C. PC and XT motherboards

1. If the PC motherboard is marked 64/256KB or the computer is an XT, the first character of the error code indicates the failing bank of chips.

0	Bank 0	(Towards rear of PC)
1	Bank 1	
2	Bank 2	
3	Bank 3	(Near front of PC)

2. The second two characters of the error code indicate the failing chip within the bank.

3. If the first number of the error code is not 0, 1, 2 or 3 then the PC may have a failure on a memory expansion board.

D. Motherboard DIP switch settings

1. The PC and XT read DIP switch settings on the motherboard to determine system configuration.

a. If an 8087 math coprocessor is installed

b. How many floppy disk drives are installed

c. How much memory is installed

d. Is CRT monitor monochrome or color

2. Check switch settings compared to actual system configuration.

3. Changing the type of video adapter without changing switch setting is a common error.

V. Power supply

A. Power supplies used in the IBM PC and clones are switching power supplies.

B. If power switch is turned off and then back on in less than six seconds, the power supply may be in shut-down condition. While this happens, the fan will not run.

C. The cooling fan on IBM products is powered from DC.

1. If the fan is running and the AC section is working, trouble could be in either the fan or a part of the DC section.

2. If the fan is not running, the problem could be in the AC input side or the DC side of the power supply. Also, the power supply could be in shut down (see B above).

D. Common power supply problems include a blown fuse, defective driver transistor and defective power output transistor.

E. Substitute NPN replacement type TO220 transistors rated at 60 watts or greater at several hundred volts for defective transistors.

VI. Keyboard

A. Common keyboard troubles include an open in the cable, sticky keys, dead keys and dirty membrane.

B. Replacement cables are available for the keyboard.

CAUTION: Keyboards contain CMOS devices; observe ESD precautions.

C. Sticky and dead keys can often be repaired.

1. Turn the keyboard bottom-up.

2. Remove bottom cover.

NOTE: For convenience, construct a wooden frame to hold the keyboard case in an inverted position with key caps hanging free.

3. Bend metal tabs to release metal plate.

4. Carefully observe position of springs and "paddles" on working keys.

5. Reseat springs and paddles or replace them with parts salvaged from a scrap keyboard.

6. Clean membrane and metal plate with alcohol and cotton swabs.

7. Reassemble and test keyboard.

8. Several attempts may have to be made to put the keyboard back together properly.

D. If the keyboard is not repairable, replace with an IBM original, an inexpensive clone import or a good-quality clone.

VII. Disk drives

A. General drive information

1. Both disk drives in the IBM and compatibles are identified by setting the drive jumper on the logic board to the second position.

2. Drives are addressed as drive A or drive B by a twist in the ribbon cable.

a. The controller card connector is on the long end of cable.

b. Drive A's connector is at the short end of the cable.

c. Drive B's connector is a few inches from the A connector.

d. A section of the cable is split and twisted at the drive A end of the cable.

3. Try this quick test when drive A does not boot.

a. Disconnect the ribbon cable from both drives A and B.

b. Connect the A ribbon connector to drive B.

c. Attempt to boot the computer.

d. Drive B is now being used as drive A.

e. If computer now boots, this proves that drive A is defective.

4. Belts on drives must be clean.

a. Do not touch the belt surface or pulley with fingers.

b. Keep lubricants from getting on belts or pulleys.

c. Skin oil or lubricants can be cleaned from belts and pulleys with alcohol.

5. It is good practice to clean and test each disk drive on every computer repair job.

B. Disk drives

1. Full-height drives for the IBM

   a. Type 1 (manufactured by Tandon--serial number prefix A, B or none)

   b. Type 2 (manufactured by Control Data--serial number prefix D)

2. Half-height drives for the IBM

   a. Type 3 (serial number prefix E)

   b. Replacement third-party half-height drive

3. Common troubles

   a. Wrong rotation speed

   b. Broken door hinge

   c. Dirty heads

   d. Stiff head position mechanism

   e. Heads out of alignment

   f. Centering cone worn

C. Speed test and adjust

1. Preliminary rotation speed can be adjusted on the Type 1 drive as follows.

   NOTE: Steps that are the same for Type 1, 2 or 3 drives are marked with an asterisk.

   a*. Remove the disk drive from system box.

   b*. Connect power cable while leaving signal cable disconnected.

   c*. Insert a blank scratch diskette into drive.

   d. Connect a jumper wire from TP-10 to TP-13 on the drive logic board.

   e*. Power up the system.

   f*. Observe the outer ring of strobe marks on the drive pulley under fluorescent lighting.

   g. Adjust the potentiometer on servo board mounted on rear of drive until strobe rings appear to stand still.

   h*. Make final speed adjustment with Advanced Diagnostics or other diagnostic disk.

2. Preliminary rotation speed can be adjusted on the Type 2 drive as follows.

   a*. Remove the disk drive from system box.

   b*. Connect power cable while leaving signal cable disconnected.

c*. Insert a blank scratch diskette into drive.

d. Connect a jumper wire from pin 16 on the drive ribbon signal connector to ground (frame of drive).

e*. Power up the system.

f*. Observe the outer ring of strobe marks on the drive pulley under fluorescent lighting.

g. Adjust the potentiometer mounted on the top left edge of logic board until strobe rings appear to stand still.

h*. Make final speed adjustment with Advanced Diagnostics or other diagnostic disk.

3. Rotation speed can be checked on the Type 3 drive as follows.

a*. Remove the disk drive from system box.

b*. Connect power cable while leaving signal cable disconnected.

c*. Insert a blank scratch diskette into drive.

d. Connect a jumper wire from pin 16 on the drive ribbon signal connector to ground (frame of drive).

e*. Power up the system.

f*. Observe the outer ring of strobe marks on the drive pulley under fluorescent lighting.

g. The speed cannot be adjusted on Type 3 and most half-height drives.

h*. Make final exact speed test with Advanced Diagnostics or other diagnostic disk.

i. If speed is out of tolerance, replace drive.

4. Adjust speed of disk drive to 2000 ±20 on Advanced Diagnostics speed test display.

D. Dirty heads

1. Magnetic heads accumulate oxides from the diskette media, which cause weak, noisy signals.

2. Follow this procedure to clean drive heads.

a. Saturate the gauze disk of a head-cleaning diskette with alcohol.

b. Insert diskette in drive.

c. Type DIR at DOS prompt to position heads against gauze.

d. Repeat several times.

e. Let heads air-dry before inserting a good diskette.

3. If head-cleaning diskettes are not available, carefully clean heads with long cotton swabs saturated in alcohol.

E. Stiff mechanism

1. "Seek" errors can sometimes be the result of dry gears and other moving parts in the head positioning assembly.

2. Lubricate sparingly with light grease or oil, such as gun slick or silicone lubricant.

F. Alignment troubles

1. To align heads and center cones, assemble the following.

a. Test equipment (expensive)

b. Service manuals (impossible to get for some models)

c. Replacement parts (impossible to get for some models)

2. If all of the listed items are not available, replace the drive.

VIII. Video displays

A. Common troubles with displays

1. Misadjusted controls

2. Blown fuse

3. Line power transformer open

4. Voltage regulator I.C. shorted

5. Horizontal output transistor blown

B. Service schematics

1. Available from Sams Computerfacts for some IBM models (See references for the unit. Sams Computerfacts are available from Howard K. Sams & Co.)

2. Not available for most replacement CRT displays

C. Adapter card replacement

1. Adapter cards are relatively inexpensive.

2. Original equipment cards can be purchased from IBM or local computer stores.

3. Import clone boards are very inexpensive and are compatible with the PC system.

4. Also, consider mail-ordering import boards from computer magazine advertisements.

BEEP INDICATORS

Beep indicator	Look for trouble at
No beep, dead PC	AC power, power supply
Continuous beep	AC power, power supply
Series of short beeps	System board
One long beep and one short beep	System board
One long beep and two shorts	Display
One short beep, blank display, incorrect display	Display
One short beep, no disk boot, cassette BASIC	Diskette, disk drive

SYSTEM ERROR CODE DISPLAY

NOTE: xxxx indicates any hexadecimal number present.

Error code	Look for trouble at
101	System board (motherboard)
201	RAM failure
xxxx=201	Memory failure
xxxx=201 PARITY CHECK x	Memory failure
PARITY ERROR 1	Memory, try seating RAM chips
301	Keyboard or cable
401	Monochrome adapter card
501	Color/graphics adapter card
601-626	Disk drive adapter card, cable, disk drive
PARITY CHECK 1 (with no other codes)	Power supply

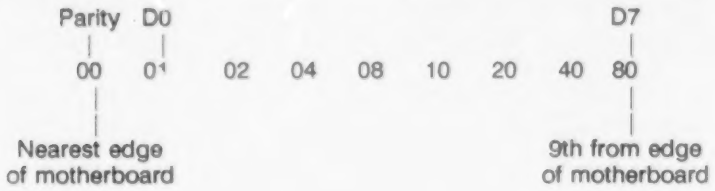
MEMORY CHIP FAILURE ERROR CODES

For PC motherboards marked 16/64KB

The first two characters of the error code indicate the failing bank of chips.

00	Bank 0	(Towards rear of PC)
04	Bank 1	
08	Bank 2	
0C	Bank 3	(Near front of PC)

The second two characters of the error code indicate the failing chip within the bank.

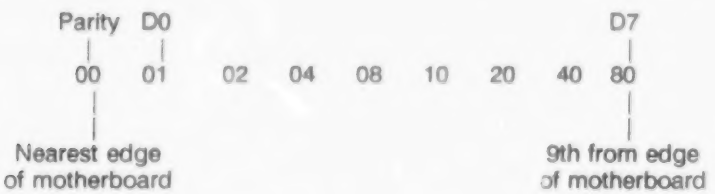


For PC motherboards marked 64/256KB or XT computers

The first character of the error code indicates the failing bank of chips.

0	Bank 0	(Towards rear of PC)
1	Bank 1	
2	Bank 2	
3	Bank 3	(Near front of PC)

The second two characters of the error code indicate the failing chip within the bank.



IBM DIP SWITCH SETTINGS

Original IBM PC five-slot motherboard

Two eight-position DIP switches are used on the motherboard to set the configuration of the motherboard.

Switch Block 1 (located near the center of the motherboard)

SW-1 Off for normal operation (See disk drive settings)

SW-2 On if using an 8087 math coprocessor
Off if no math coprocessor

SW-3, SW-4 System Board Memory

SW-3 On SW-4 Off	

SW-5, SW-6 Use of display adapter

SW-5 On, SW-6 On	No display adapter
SW-5 Off, SW-6 Off	Monochrome display adapter
SW-5 Off, SW-6 On	Color graphics adapter (40 x 25 mode)
SW-5 On, SW-6 Off	Color graphics adapter (80 x 25 mode)

SW-1, SW-7, SW-8 Use of disk drives

SW-1 On, SW-7 On, SW-8 On	No disk drive
SW-1 Off, SW-7 On, SW-8 On	One disk drive
SW-1 Off, SW-7 Off, SW-8 On	Two disk drives

Switch Block 2 (located near the edge of the motherboard)

SW-1, SW-2, SW-3, SW-4, SW-5 Amount of memory installed

SW-1 On, SW-2 On, SW-3 On, SW-4 On, SW-5 On	64K total memory
SW-1 On, SW-2 Off, SW-3 On, SW-4 On, SW-5 On	128K total memory
SW-1 On, SW-2 On, SW-3 Off, SW-4 On, SW-5 On	192K total memory
SW-1 On, SW-2 Off, SW-3 Off, SW-4 On, SW-5 On	256K total memory

SW-6, SW-7, SW-8 Always Off

Eight-slot motherboards

The single eight-position DIP switch of the motherboard is used to set the configuration of the motherboard.

SW-1 Off for normal operation

SW-2 Off if using an 8087 math coprocessor
On if no math coprocessor

SW-3 System board memory
SW-4

SW-3 Off, SW-4 On	128K
SW-3 On, SW-4 Off	192K
SW-3 Off, SW-4 Off	256K

SW-5 Use of display adapter
SW-6

SW-5 On, SW-6 On	No display adapter
SW-5 Off, SW-6 On	Color graphics adapter (40 x 25 mode)
SW-5 On, SW-6 Off	Color graphics adapter (80 x 25 mode)
SW-5 Off, SW-6 Off	Monochrome display adapter

SW-7 Use of disk drives
SW-8

SW-7 On, SW-8 On	One disk drive
SW-7 Off, SW-8 On	Two disk drives
SW-7 On, SW-8 Off	Three disk drives
SW-7 Off, SW-8 Off	Four disk drives

XT turbo clone

The single eight-position DIP switch of the motherboard is used to set the configuration of the motherboard.

SW-1 Off for normal operation

SW-2 On if using an 8087 math coprocessor
Off if no math coprocessor

SW-3 Memory usually used with jumper wires on the
SW-4 motherboard. See motherboard manual.

SW-5 Use of display adapter
SW-6

SW-5 On, SW-6 On	No display adapter
SW-5 Off, SW-6 On	Color graphics adapter (40 x 25 mode)
SW-5 On, SW-6 Off	Color graphics adapter (80 x 25 mode)
SW-5 Off, SW-6 Off	Monochrome display adapter

SW-7 Use of disk drives
SW-8

SW-7 On, SW-8 On	One disk drive
SW-7 Off, SW-8 On	Two disk drives
SW-7 On, SW-8 Off	Three disk drives
SW-7 Off, SW-8 Off	Four disk drives

RUN IBM ADVANCED DIAGNOSTICS

Equipment:

IBM PC or compatible
Advanced diagnostic diskette
Blank, formatted, "scratch" diskette
Wrap plugs if using the IBM test package

Procedure:	Evaluation 3	2	1	N
1. Boot advanced diagnostics.	—	—	—	—
2. Confirm PC configuration.	—	—	—	—
3. Test motherboard.	—	—	—	—
4. Test memory.	—	—	—	—
5. Test video adapter and CRT.	—	—	—	—
6. Test keyboard.	—	—	—	—
7. Test printer port.	—	—	—	—
8. Test floppy drive A.	—	—	—	—
9. Test floppy drive B.	—	—	—	—
10. Test PC.	—	—	—	—
11. Analyze diagnostic display. Discuss diagnostics with instructor in a knowledgeable manner.	—	—	—	—
12. How long did it take to perform these steps? (_____ hrs., _____ min.)	—	—	—	—
13. Turn equipment off; store equipment and tools properly.	—	—	—	—
Average of above evaluations*	—	—	—	—

*This average is the student's final evaluation for competency I9: Troubleshoot hardware problems.

TEAR DOWN, REASSEMBLE AND TEST A PC

Equipment:

IBM PC or compatible
Advanced diagnostic diskette
Blank, formatted, "scratch" diskette
Hand tools
ESD anti-static mat and bracelet

NOTE: Observe how all modules interconnect. Then note, draw or mark with numbered tape so that the modules and cables can be properly reassembled later.

Procedure:		Evaluation 3	2	1	N
1.	Run diagnostics on PC to verify proper operation.	—	—	—	—
2.	Turn PC power off. Unplug power cord, keyboard and CRT. Remove cover, touching the power supply cage to ground yourself.	—	—	—	—
3.	Lay out ESD mat. Attach bracelet, ESD cables and clip to PC power supply cage.	—	—	—	—
	NOTE: As boards and modules are removed, lay them on the ESD mat.				
4.	Remove adapter boards from card slots.	—	—	—	—
	NOTE: Carefully observe how cables attach to floppy-disk controller card.				
5.	Remove floppy disk drives.	—	—	—	—
	NOTE: Observe carefully how cables attach to diskette drives. The connector on the end is on drive A.				
6.	Unplug power supply cables from motherboard.	—	—	—	—
7.	Remove power supply.	—	—	—	—
8.	Unplug and remove loudspeaker.	—	—	—	—
9.	Remove screws holding motherboard to chassis.	—	—	—	—
10.	Slide motherboard out of case.	—	—	—	—
11.	Reassemble PC modules and cables (in reverse order).	—	—	—	—
12.	Attach keyboard, power cord and CRT.	—	—	—	—
13.	Run diagnostics to verify proper reassembly.	—	—	—	—
14.	Ask the instructor to check for correct tear-down and reassembly of the PC.	—	—	—	—

15.	Ask the instructor to check for correctly analyzed diagnostic display. Discuss tear-down and reassembly with the instructor in a knowledgeable manner.	—	—	—	—
16.	Replace cover on PC case.	—	—	—	—
17.	Run diagnostics to verify proper reassembly.	—	—	—	—
	NOTE: Always run diagnostics again after cover is installed.				
18.	How long did it take to perform these steps? (_____ hrs., _____ min.)	—	—	—	—
19.	Turn equipment off; store equipment and tools properly.	—	—	—	—
Average of above evaluations*		—	—	—	—

*This average is the student's final evaluation for competency I9: Troubleshoot hardware problems.

TROUBLESHOOT AND REPAIR A DEFECTIVE PC

Equipment:

IBM PC or compatible with spare boards and modules
Advanced diagnostic diskette and "scratch" diskette
Hand tools
ESD anti-static mat and bracelet
Service manuals

NOTE: This job sheet can be copied and repeated for additional troubleshooting jobs as required by the instructor.

Procedure:	Evaluation 3	2	1	N
1. Run diagnostics on PC to determine troublesome area.	—	—	—	—
2. Disassemble PC.	—	—	—	—
3. Observe correct ESD precautions.	—	—	—	—
4. Locate defective unit or module.	—	—	—	—
5. Replace defective unit or module.	—	—	—	—
6. Run diagnostics to verify repair.	—	—	—	—
7. Ask the instructor to check for correctly analyzed diagnostic display. Discuss the troubleshooting job with the instructor in a knowledgeable manner.	—	—	—	—
8. Replace cover on PC case.	—	—	—	—
9. Run diagnostics to verify proper reassembly.	—	—	—	—
NOTE: Always run diagnostics again after cover is installed.				
10. How long did it take to perform the repair? (_____ hrs., _____ min.)	—	—	—	—
11. Turn equipment off; store equipment and tools properly.	—	—	—	—
12. Make out repair bill, if required.	—	—	—	—
Average of above evaluations*	—	—	—	—

*This average is the student's final evaluation for competency I9: Troubleshoot hardware problems.